



सत्यमेव जयते

GUIDELINES FOR DEVELOPMENT OF e-GOVERNANCE APPLICATIONS (GuDApps)



Digital India
Power To Empower



**NATIONAL
INFORMATICS
CENTRE**



**GUIDELINES
FOR
DEVELOPMENT OF e-GOVERNANCE
APPLICATIONS (GuDApPs)**

**GUIDELINES
NIC-GDL-DA-1.1**

AUGUST 2017

रविशंकर प्रसाद
RAVI SHANKAR PRASAD



मंत्री
विधि एवं न्याय
एवं
इलेक्ट्रॉनिकी और सूचना प्रौद्योगिकी
भारत सरकार
MINISTER OF
LAW & JUSTICE
and
ELECTRONICS & INFORMATION TECHNOLOGY
GOVERNMENT OF INDIA


MESSAGE

With the launch of the Digital India Programme, we have embarked on a journey to transform India into a digitally empowered society and knowledge economy. All the departments in the Central, State and local governments are fully geared to fulfil this vision. Governments at all levels are relying more and more on eGovernance solutions for performing government functions and providing services to citizens and businesses. Data captured by these solutions is becoming the most critical asset of the Government enabling powerful analytics and driving informed, data-driven decision making.

The importance of quality eGovernance solutions cannot be overstated. Quality and standards based solutions that are built in a rapid development environment is the need of the hour. This document is a timely compilation of the guidelines and best practices which will enable the architects and developers in quickly developing quality solutions that conform to software as well as data quality.

NIC, with its vast experience as an eGovernance solution consultant and provider is rightly positioned to publish this document and I congratulate NIC for this excellent effort. I am sure this would go a long way in creating awareness and facilitate application development teams to make better, smarter and efficient solutions.

I am sure NIC will sustain this effort and take it forward to address all aspects of eGovernance solutions so that the vision of the Digital India is fully realized.



(RAVI SHANKAR PRASAD)

पी.पी. चौधरी
राज्य मंत्री
विधि एवं न्याय
और
इलेक्ट्रॉनिकी और सूचना प्रौद्योगिकी
भारत सरकार



P.P. CHAUDHARY
Minister of State
Law & Justice
and
Electronics & Information Technology
Government of India

Message

India has embarked on a journey to transform governance that is committed to quality, efficiency and accessibility. eGovernance solutions have transformed the landscape of governance by enabling us to realize this vision. As Government's reliance on eGovernance solutions increases by the day, the need for quality solutions that are deployed at an increasing pace is becoming more and more important.

If we have to realize the dream of a connected government, then it becomes imperative that applications are developed in conformity to certain standards and guidelines. I congratulate the NIC team for the effort they have put in developing guidelines and best practices using their years of experience, which would go a long way in creating awareness and facilitate application development teams to make better, smarter and more efficient solutions.

While the Government of India is actively promoting the Digital India campaign to facilitate maximum governance through IT empowerment of citizen, the quality and reliability of all eGovernance projects need to rise up to the desired quality standard and in this context, this initiative will contribute its worth.

There can be no doubt that these guidelines and best practices will achieve their purpose of improving data quality and efficiency of application development process. It is anticipated that the document will be regularly revised and updated in response to latest developments in technology.

(P.P. Chaudhary)





अजय साहनी, आई.ए.एस.
AJAY SAWHNEY, I.A.S.

सचिव
इलेक्ट्रॉनिकी और सूचना प्रौद्योगिकी मंत्रालय
भारत सरकार
Secretary
Ministry of Electronics &
Information Technology (MeitY)
Government of India

FOREWORD

With increased dependency of Government on Data for policy and decision making, the development of quality eGovernance solutions has become crucial to ensure right decisions. NIC has made efforts to establish standards to streamline software processes for development of e-governance applications with consistent interfaces through the **Guidelines for the Development of Good Applications (GuDApps)**.

I hope the GuDApps Guidelines will be helpful to facilitate application development teams to develop quality applications as well as ensuring quality of data captured by them.

I would like to compliment NIC on taking this proactive step and look forward to more such best practices and guidelines to quickly build quality software solutions and develop this culture by spreading awareness and training.

In our endeavor to promote data quality and accountability through increased use of technology and online services, I would like to request the users to provide their valuable feedback based on their experience and requirements for continuous enhancements of these Guidelines as quality is not a destination, but a journey.


(Ajay Sawhney)

नीता वर्मा
महानिदेशक

Neeta Verma
Director General



भारत सरकार
इलेक्ट्रॉनिक्स और सूचना प्रौद्योगिकी मंत्रालय
राष्ट्रीय सूचना-विज्ञान केन्द्र
ए-ब्लॉक, केन्द्रीय कार्यालय परिसर, लोधी रोड
नई दिल्ली-110 003 (भारत)

Government of India
Ministry of Electronics and Information Technology
National Informatics Centre
A-Block, CGO Complex, Lodhi Road
New Delhi-110 003 (India)
Tel : +91-11-24361504, +91-11-24361447
Fax : +91-11-24364873
E-Mail : dg@nic.in Website : www.nic.in

PREFACE

Government at all levels are increasingly relying on digital eGovernance solutions for performing government functions and providing services to citizens and businesses. In order to establish an efficient and effective e-Governance ecosystem, it is important that e-Governance solutions are robust, inter-operable, intelligent and user-friendly. Data security and confidentiality is another critical aspect that will impact adoption of such systems.

This document is the much needed attempt to define guidelines and best practices for architecting eGovernance solutions and shall play a significant role in building robust, efficient, and high quality software solutions. The guidelines address all important aspects of development of secure and standards compliant ICT solutions for citizen services.

Guidelines include recommendations on data quality, user authentication, electronic forms, document management, reports design, reporting framework, and application development frameworks including design patterns and parameters. As software development frameworks play a crucial role in faster development of applications that are standard, user-friendly, replicable & offer consistent user interfaces for data capture and presentation, has been adequately covered in the guideline document.

It is hoped that software professionals developing eGovernance solutions will find it as a ready reference guide. It may also be useful for government practitioners while defining the specifications for eGov applications. As a living document, it is expected to grow both in breadth and depth as it is put to use by the professionals and other dimensions of software quality improvement are identified.



(Neeta Verma)

ABSTRACT

This document provides guidelines for building quality e-Governance applications. In particular, the guidelines address the issues related to data quality parameters, user interface, report design, authentication and software frameworks. It is expected that the use of these guidelines will lead to better data quality and faster delivery of robust e-Governance applications.

Keywords: Guidelines, Data Quality, e-Governance Applications, Software Quality, Document Storage, Authentication, Forms, Reports, Frameworks

Government of India
Ministry of Electronics & Information Technology
National Informatics Centre
New Delhi

August, 2017

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of NIC.

Contributors

Prepared by	Smt. Alka Mishra, Scientist F, NIC Sh. Joydeep Shome, Scientist F, NIC Sh. Pawan Kumar Joshi, Scientist F, NIC Smt. Rachna Srivastava, Scientist F, NIC Sh. Rajender Sethi Scientist F, NIC Smt. Rama Hariharan, Scientist F, NIC Dr. Rajesh Kumar Mishra, Scientist E, NIC
Reviewed By	Sh. Girish Kumar Gaur, Scientist G, NIC Dr. (Smt.) Savita Dawar, Scientist F, NIC
Approved By	Smt. Neeta Verma, Director General, NIC

Amendment log

Version	Date	Brief Description	Section Change
1.0	2nd May 2017	The document focuses on guidelines and best practices for ensuring quality of data captured in e-Governance solutions covering data, authentication, forms/reports, development frameworks.	
1.1	29th August 2017	The acronyms/abbreviations, overview has been revised and references has been shifted to Appendix.	1.3, 1.4, 1.5, Appendix I

CONTENTS

01	Introduction	01
1.1	Purpose	
1.2	Scope	
1.3	Definitions, Acronyms and Abbreviations	
1.4	Overview	
02	Data Quality	09
2.1	Data Element	
2.1.1	Data Identification	
2.1.1.1	Data Element Name	
2.1.1.2	Aliases	
2.1.1.3	Description	
2.1.1.4	Data Source	
2.1.1.5	Base or Derived	
2.1.1.6	Privacy and Security	
2.1.2	Data Size	
2.1.2.1	Data Type	
2.1.2.2	Data Length	
2.1.3	Data Domain	
2.1.3.1	Acceptable Values	
2.1.3.2	Default Values	
2.1.3.3	Mandatory or Optional	
2.1.4	Validations	
2.1.5	Verification	
2.1.6	Data Availability	
2.1.7	User Interface	
2.1.7.1	Data Input	
2.1.7.2	Generic Interface Guidelines	
2.1.7.3	Input using List of Values	
2.1.7.4	Input using Search	
2.1.7.5	Field Caption	
2.1.7.6	Output Format	
2.1.8	Metadata Standards	
2.2	Record Element	
2.2.1	Record Identification	
2.2.2	Record Level Validation	
2.3	Data Functions	
2.3.1	Data Function/Table Identification	
2.3.1.1	Table Name	
2.3.1.2	Primary Key for a Table	

CONTENTS

2.3.2	Referential Integrity
2.4	Identifiers
2.4.1	Criteria for Defining of New Identifier
2.4.2	Common Identifiers
2.4.2.1	Aadhaar
2.4.2.2	Permanent Account Number (PAN)
2.4.2.3	District Information System for Education (DISE) Code
2.4.2.4	Labour Identification Number (LIN)
2.4.2.5	Indian Financial System Code (IFSC)
2.5	Guidelines for Common Data Elements

03 Authentication

25

3.1	Authentication Levels
3.1.1	Single-factor authentication
3.1.2	Two-factor authentication [2FA]
3.1.3	Multi-factor authentication
3.2	Authentication Types
3.2.1	HTTP Basic Authentication
3.2.2	Form Based Authentication
3.2.3	Digital Certificates (SSL and TLS)
3.2.4	One Time Password
3.2.5	Biometric Authentication
3.3	Implementation of Authentication
3.3.1	HTTP Basic/ Form Based Authentication
3.3.2	Authentication using Database
3.3.3	Authentication using LDAP
3.3.4	Certificate based Authentication
3.3.5	One Time Password Based Authentication
3.3.6	Custom Application generated OTP
3.3.7	Aadhaar Based OTP
3.3.8	Time Based One Time Password (TOTP)
3.3.9	Biometric Based Authentication
3.4	Sign-up/login Processes
3.4.1	Ways to Sign-up
3.4.1.1	Sign-up using Application Specific User-id/Password
3.4.1.1.1	On-line Sign-up
3.4.1.1.1.1	Confirm Email for Sign-up

CONTENTS

- 3.4.1.1.1.2 Set User-id/Password (Confirm Mobile)
- 3.4.1.1.2 Off-line Sign-up
- 3.4.1.1.3 Bulk Sign-up
 - 3.4.1.1.3.1 Bulk Sign-up using Name, Address
 - 3.4.1.1.3.2 Bulk Sign-up using Email and Mobile number
- 3.4.1.2 Sign-up using Official User-id/Password (e.g. NIC Email)
- 3.4.1.3 Sign-up using Social Networking User-id/Password
- 3.4.2 Ways to recall Sign-up Credentials
 - 3.4.2.1 Users with Email/Mobile
 - 3.4.2.1.1 Forget User-id
 - 3.4.2.1.1.1 Know User-id
 - 3.4.2.1.2 Forget password
 - 3.4.2.1.2.1 Set Password
 - 3.4.2.2 Users without Email/Mobile
 - 3.4.2.2.1 Reset User-id/Password
- 3.4.3 Ways to Login
- 3.4.4 Ways to Change/Deactivate Credentials
 - 3.4.4.1 Change Password
 - 3.4.4.2 Submit Email De-activation Request
 - 3.4.4.2.1 De-activate Email Address
 - 3.4.4.3 Submit Mobile De-activation Request
 - 3.4.4.4 Deactivate User-id
- 3.5 Additional Best Practices
 - 3.5.1 Stop Auto User Creation
 - 3.5.2 Using Captcha
 - 3.5.3 Context Based Authentication
 - 3.5.4 Additional image based profile verification
 - 3.5.5 Using Forgot Password
 - 3.5.5.1 Reset password link
 - 3.5.5.2 Temporary Password
 - 3.5.6 Using Profile/Transactional Password
 - 3.5.7 Security Questions
 - 3.5.8 New Account Activation links
 - 3.5.9 Account Locking
 - 3.5.10 Account Audit Policy
- 3.6 Other Guidelines & Conclusion

CONTENTS

04 Forms

45

- 4.1 User-Centric Approach
 - 4.1.1 Relationship
 - 4.1.2 Conversation
 - 4.1.3 Appearance
- 4.2 Structuring of Form
- 4.3 Form Elements
 - 4.3.1 Labels
 - 4.3.2 Input Fields
 - 4.3.2.1 Text Boxes
 - 4.3.2.2 Radio Buttons
 - 4.3.2.3 Check Boxes
 - 4.3.2.4 Dropdown Lists & Combo boxes
 - 4.3.2.5 List Boxes
 - 4.3.3 Actions
 - 4.3.4 Help Text
 - 4.3.4.1 Form level Instructions
 - 4.3.4.2 Inline Instructions
- 4.4 Form Validations
 - 4.4.1 Validation Methods
 - 4.4.1.1 Server-side validation
 - 4.4.1.2 Client-side validation
 - 4.4.2 Validation Types
 - 4.4.3 Validation Feedback
- 4.5 Document Upload, Storage and Management
 - 4.5.1 Document Upload
 - 4.5.1.1 Typical Use-Case Scenario
 - 4.5.1.2 Issues and Concerns to be Addressed
 - 4.5.1.3 Functional and Operational issues in File Upload Process
 - 4.5.1.3.1 Advisories Regarding Image File Upload
 - 4.5.1.3.2 Restrictions on Document Type and Extension
 - 4.5.1.3.3 Restriction on File Size
 - 4.5.1.3.3.1 Size restrictions for image file
 - 4.5.1.3.3.2 File Resizing
 - 4.5.1.4 Issues to be Addressed : Security Vulnerabilities
 - 4.5.1.4.1 Typical forms of malicious attack
 - 4.5.1.4.2 Consequences of a malicious attack
 - 4.5.1.4.3 Defending against file upload attacks

CONTENTS

- 4.5.2 Storage Options : Database v/s File System
 - 4.5.2.1 Relational DB
 - 4.5.2.2 File System
 - 4.5.2.3 NoSQL DB
- 4.5.3 Document Management

05 Reports

79

- 5.1 Introduction
- 5.2 Conduct A Thorough Analysis Of User Scenarios
- 5.3 Know Your User
- 5.4 Setting Up A Report – The Query Filter
- 5.5 Report Layout
- 5.6 Emphasize Important Information
- 5.7 Format And Paginate
- 5.8 Make The Report Distributable
- 5.9 Design Database Specifically For Reports
 - 5.9.1 Transactional Vs Reporting Database
 - 5.9.2 When To Build A Separate Reporting Database?
 - 5.9.3 Advantages Of Having A Separate Reporting Database
 - 5.9.4 Best Practices For Building A Reporting Database
- 5.10 Reporting Frameworks
- 5.11 Conclusion

06 Application development Frameworks

95

- 6.1 Framework Design Pattern
 - 6.1.1 Inversion of control
 - 6.1.2 Extensibility
 - 6.1.3 Non-modifiable framework code
- 6.2 Framework Components/Parameters
 - 6.2.1 Packages/Wrappers
 - 6.2.2 Architecture
 - 6.2.2.1 Design Pattern
 - 6.2.2.1.1 Model-View-Controller
 - 6.2.2.1.2 Push-based vs. pull-based
 - 6.2.3 Methodology
- 6.3 Java based Frameworks

CONTENTS

- 6.3.1 Apache Struts
- 6.3.2 Spring
- 6.3.3 JSF
- 6.4 PHP Based Framework
- 6.4.1 Most Popular PHP Frameworks
 - 6.4.1.1 Laravel
 - 6.4.1.2 Symfony
 - 6.4.1.3 CodeIgniter
 - 6.4.1.4 CakePHP
 - 6.4.1.5 Slim
- 6.4.2 PHP Frameworks Summary

Appendix

105

- A. Office Orders
- B. Code Directories
- C. Common Data Elements
- D. Person Specific Data Elements
- E. Geo References Data Elements
- F. Labour & Employment Data Elements
- G. Compliance Matrix
- H. Case Study for Compliance Matrix on Data Quality
- I. References

1.1 PURPOSE

1.2 SCOPE

1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

1.4 OVERVIEW

INTRODUCTION

The expectations of government to deliver more services and to deliver them better presents a challenge for NIC. Well-engineered automated solutions can only increase productivity in service delivery to help in meeting these expectations.

A Technical Advisory Group was setup with an objective to develop enterprise software solutions quickly by re-using what is already available without compromising the quality and reinventing the wheel.

In this context, guidelines have been prepared to arrive at common data standards, interface, and report design, authentication and security, application framework and so on. If used properly, this document can act as the guiding document for all major ICT projects taken up by NIC, so as to ensure an integrated, coordinated and standards-based effort. In the absence of such a uniting fabric, it is likely that there would be avoidable duplication, incompatibility, delay and inefficiency in delivery of efficient ICT projects.

The guideline aims to streamline software processes for development of e-governance applications with consistent interfaces. As such guidelines are not mandatory and the purpose of these guidelines is to provide a set of practices to make development efforts more predictable and presumably of higher quality in terms of the system interactions or interfaces with its external environment.

1.1

Purpose

For web sites/ web applications, the GIGW (“Guidelines for Government Websites”) are put in place. But there are no guidelines/best practices available for standardization of application architecture, User Interface, validations and data architecture. This has resulted in applications which are not reusable and interoperable. A need is felt to develop standard guidelines for software development projects carried out by NIC.

The purpose is to document set of practices towards data quality which can be used to save on development efforts to keep the consistency for end-user with predictable system interactions with its external environment. The proposed guidelines will take care for data, its user interfaces and data quality parameters. The guidelines will be referred as Guidelines for Development of e-Governance Applications (GuDApps).

1.2

Scope

The scope of this document is to focus on various components of assured quality for e-governance applications covering the following aspects

- **Data Quality**

Includes data dictionary, validation/verification, availability and its presentation; their grouping and referential integrity; identifiers; reference/code directories.

- **Authentication**

Authentication levels, types, implementation and best practices

- **Forms and Reports design**

Includes design principles, form structure, reporting frameworks, validation/verification, document upload and storage guidelines, security etc.

- **Application Development Framework**

Design patterns, components, parameters, Java and PHP Frameworks

1.3

Definitions, Acronyms and Abbreviations

The definitions listed below establish meaning in the context of this requirement specification.

Item	Description
2FA	Two Factor authentication
ANSI	American National Standards Institute
AOP	Aspect Oriented Programming
API	Application Programming Interface
Authorised Representative	Any registered user who has been authorised by Employer Organisation to submit claims, file objections and use the web application for various activities on its behalf.
BLOB	Binary Large Objects

Item	Description
BSON	Binary JSON
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CR, LF	Carriage Return, Line Feed
CSV	Comma Separated Values
DISE	District Information System for Education
DOS	Denial of Service
ECI	Election Commission of India
End User	The person or persons who will ultimately be using the system for its intended purpose.
EPFO	Employees Provident Fund Organisation
ETL	Extract, Transfer and Load
GIGW	Guidelines for Government Websites
HMAC	Hashed message authentication code
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technologies
IFSC	Indian Financial System Code
IOC	Inversion of Control
JAX-RS	Java API for RESTful Web Services
JPEG	Joint Photographic Experts Group
JSF	Java Server Faces
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
LIN	Labour Identification Number
MDDS	Meta data and data standards
MIME	Multipurpose Internet Mail Extensions
MVC	Model View Controller

Item	Description
NIC	National Informatics Centre
OATH	OATH is an industry-wide collaboration to develop an open reference architecture by leveraging existing open standards for the universal adoption of strong authentication.
OLAP	On-line Analytical Processing
OLTP	On-line Transaction Processing
ORM	Object Relational Mapping
OTP	One Time Password
OWASP	Open Web Application Security Project
PAN	Permanent Account Number
Password	Password associated with the username defined above to be checked for authentication
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PIN	Personal identification Number
PKI	Public Key Infrastructure
PNG	Portable Network Graphics format
Registered User	Any person who registers with the web application.
REST	Representational State Transfer
RFC	Request for comments
SaaS	Software as a service
SASL	Simple Authentication and Security Layer
SDLC	Software Development Life Cycle
SHA	Secure Hash Algorithm
SMS	Short Message Service
SQL	Structured Query Language

Item	Description
SRS	Software Requirements Specification documents essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics
SyRS	System Requirement Specification provides a “black-box” description of what the system should do, in terms of the system interactions or interfaces with its external environment.
TIFF	Tagged Image File Format
TLS	Transport Layer Security
TOTP	Time Based One Time Password
UIDAI	Unique Identification Authority of India
URL	Uniform Resource Locator
Username	The personal identity of the user used for authentication, It may be user-id linked with LDAP or Stored in the database
Verhoeff Algorithm	It was the first decimal check digit algorithm which detects all single-digit errors, and all transposition errors involving two adjacent digits,
VPN	Virtual Private Network
XML	Extensible Mark-up Language
XSS	Cross-site Scripting

1.4

Overview

The following section explains data quality requirements at data elements and its parameters like identification, size/type, domain, validation/verification, its availability, kind of user interface and standards that may be followed. It also touches upon record identification/validation at record level, data functions and provides criteria for defining a new identifier, discusses some of the identifiers and their structure to understand pros and cons faced during implementation and provides information about common data elements.

The various aspects of authentication levels/ways in which someone may be authenticated using single-factor/two-factor or multi-factor verification, five type of authentication (like HTTP, form, OTP, DSCs, Biometric) & their implementation and processes/use-cases along with best practices are explained in subsequent section.

The section on Forms talks about form design principles, their structure, its elements (like labels, fields, action, help) and best practices for web form validations. The various scenarios, issues/concerns related to document upload and storage have been addressed along with storage options and document management. The section also provides a checklist which can be used for validating the designed form.

The Form section is followed by section on reporting aspects like analysing the report requirement like who will be using the report, what parameters it should have etc., report categories like MIS/Monitoring/Performance or Exceptional, designing/layout, its distribution i.e. through email or print or web-only etc. This section also discusses requirement for a separate reporting database, its advantages and best practices for building reporting database.

The section on development frameworks provides information about available java/php based frameworks for faster delivery of robust applications.

The guideline provides a way to assess the level of its compliance as per checklist given in Appendix Compliance Matrix. The various sections of the guideline as mentioned in the Compliance Matrix can be treated as recommendations of the guideline. A case study has also been given to arrive at a value in percentage of compliance in Appendix on Case Study for Compliance Matrix.

CH 02

DATA QUALITY

- 2.1 DATA ELEMENT**
 - 2.1.1 DATA IDENTIFICATION**
 - 2.1.1.1 DATA ELEMENT NAME**
 - 2.1.1.2 ALIASES**
 - 2.1.1.3 DESCRIPTION**
 - 2.1.1.4 DATA SOURCE**
 - 2.1.1.5 BASE OR DERIVED**
 - 2.1.1.6 PRIVACY AND SECURITY**
 - 2.1.2 DATA SIZE**
 - 2.1.2.1 DATA TYPE**
 - 2.1.2.2 DATA LENGTH**
 - 2.1.3 DATA DOMAIN**
 - 2.1.3.1 ACCEPTABLE VALUES**
 - 2.1.3.2 DEFAULT VALUES**
 - 2.1.3.3 MANDATORY OR OPTIONAL**
 - 2.1.4 VALIDATIONS**
 - 2.1.5 VERIFICATION**
 - 2.1.6 DATA AVAILABILITY**
 - 2.1.7 USER INTERFACE**
 - 2.1.7.1 DATA INPUT**
 - 2.1.7.2 GENERIC INTERFACE GUIDELINES**
 - 2.1.7.3 INPUT USING LIST OF VALUES**
 - 2.1.7.4 INPUT USING SEARCH**
 - 2.1.7.5 FIELD CAPTION**

2.1.7.6	OUTPUT FORMAT
2.1.8	METADATA STANDARDS
2.2	RECORD ELEMENT
2.2.1	RECORD IDENTIFICATION
2.2.2	RECORD LEVEL VALIDATION
2.3	DATA FUNCTIONS
2.3.1	DATA FUNCTION/TABLE IDENTIFICATION
2.3.1.1	TABLE NAME
2.3.1.2	PRIMARY KEY FOR A TABLE
2.3.2	REFERENTIAL INTEGRITY
2.4	IDENTIFIERS
2.4.1	CRITERIA FOR DEFINING OF NEW IDENTIFIER
2.4.2	COMMON IDENTIFIERS
2.4.2.1	AADHAAR
2.4.2.2	PERMANENT ACCOUNT NUMBER (PAN)
2.4.2.3	DISTRICT INFORMATION SYSTEM FOR EDUCATION (DISE) CODE
2.4.2.4	LABOUR IDENTIFICATION NUMBER (LIN)
2.4.2.5	INDIAN FINANCIAL SYSTEM CODE (IFSC)
2.5	GUIDELINES FOR COMMON DATA ELEMENTS

DATA QUALITY

The era of Data Driven Governance is dawning. With increasing reliance by Government on the data captured by e-Governance solutions for major policy decisions, the data quality has assumed prime importance. Data today is a critical asset of the government. It is increasingly driving the policy decisions, benefits are transferred to beneficiaries based on data captured by e-Governance applications and performance of government functionaries and field offices and of the government itself is being judged based on the data captured by e-Governance applications. It is, therefore, imperative that sufficient attention is given to ensuring quality of data captured by these applications.

In this context, this section provides guidelines to arrive at common data standards and interfaces to be used to take care for data, its user interfaces and data quality parameters.

2.1

Data Element

This section deliberates on various attributes of data element with an objective to provide clear, comprehensive information about the data that make up the system. To ensure data quality, one needs to take care these attributes at various stages of SDLC i.e. at the time of requirement gathering, database design, user interface, and its implementation at application architecture level.

A data element, also called a data item or field, is the smallest piece of data that has meaning, which need not be broken further. Often a data element is self defining such as Student name, enrolment number. The attributes of data element includes name/label, type, length, output format etc.

2.1.1 DATA IDENTIFICATION

2.1.1.1 Data Element Name

Name of the data element should be meaningful to users and clear at the database level. The name of the element should be unique in the data dictionary to avoid any ambiguity. A name such as Year or Date means nothing; the name should be contextualized depending on the domain.

A naming convention should be adopted to enhance clarity, consistency, meaningful data and better understanding among team members. In absence of naming conventions, it is time consuming to understand the data structure within team or otherwise. It becomes difficult even to debug. Two popular naming conventions are given below:

- Camel Case - Spaces and punctuation are removed.
 - o Upper Camel Case (CamelCase)

The first letter of each word is capitalised. For example ApplicantName, ScholarshipScheme, ParentOccupation

- o Lower Camel Case (camelCase)

The first letter of first word is kept small and first letter of other words is capitalised. For example applicantName, scholarshipScheme, parentOccupation

- Snake Case - Punctuation is removed and spaces are replaced by single underscores. Normally the letters share the same case.
 - o Upper Case Embedded Underscore

For example APPLICANT_NAME, SCHOLARSHIP_SCHEME, PARENT_OCCUPATION

- o Lower Case Embedded Underscore

For example applicant_name, scholarship_scheme, parent_occupation

It needs to be ensured that spelling of the name chosen should be correct one and avoid using abbreviation. For example aadhaar_number should not be written as adhar_no or aadhar_no or aadhaar_no or aadhar_number etc.

The recommendations are

1. Name of the data element should be meaningful to users and clear at the database level.
2. A naming convention should be adopted consistently across an application.
3. The spelling of name should be correct.

2.1.1.2 Aliases

Aliases are names used by different users within different systems. For example, EPFO Registration Number may be called as EPFO Number or EPF Regn Number; a Customer Number may be called a Receivable Account Number, Client Number

2.1.1.3 Description

This is an extension of the previous parameter. A clear cut description of what the data item describes should be recorded. The descriptive information may indicate its purpose, possible validation rules, default values etc.

2.1.1.4 Data Source

The data source should be known for the data elements values, the source could be a specific form, a department or outside organization, another information system or the result of a calculation.

2.1.1.5 Base or Derived

A base element is one that has been initially keyed into the system. A derived element is one that is created by a process, usually as the result of a calculation or some logic. If the data element value is the result of a calculation, then document the formula for the data element, including significant digits and rounding operations; if any

For example, based on Joining Date (i.e. base element), Relieving Date (again a base element), Leave Type, a working status is maintained as derived element; based on date of birth (i.e. base element), age (derived element) is calculated on the day of processing.

2.1.1.6 Privacy and Security

The identification for the individual or department that has access or update privileges for each data elements and identification of the users responsible for entering and changing values for the data elements should be maintained in order to maintain privacy and security. For example, date of birth is critical information which can be used as one of the parameters for identification, so should not be shown in public.

2.1.2 DATA SIZE

2.1.2.1 Data Type

Data Type should ideally be defined based on the type of data values that the data element will take. Choice of data type should also be governed by what data type is optimal for the data element in terms of minimum storage, ease of retrieval and the nature of its usage. Type refers to whether the data element contains numeric, integer, char, date values.

In case, the data element is going to be an Identifier with fixed length, it should be Char only. For example Aadhaar Number comprises of 12 digits, but is not declared as Numeric or Integer. The same should have been done with State Code. In such case, it is also to be ensured that 1st digit is not taken as zero, as while taking data from other sources like excel, it may treat is as numeric and suppress zero resulting in one digit from States with codes from 1 to 9.

In case of data element is going to be part of any identifier, it is again to be ensured that its type is Char. For example State_Code has data type as Integer, so the value from 1 to 9 will be of single character; for 10 to 36, it will be of 2 digits. This issue came up in DISE code (comprising of 11 characters with 1st two digits as State Code), the validation

of 11 character failed for States having code from 1 to 9 as it made the DISE code of 10 characters or one need to prefix zero before such States.

For example, If a data element can take only numeric values, then it may be more appropriate to use a numeric data type such as Integer or float if some calculations or numeric operations are to be performed on the data element; if, however, it is stored for displaying in reports etc or if some string operations are to be performed, then it would be more ideal to store as a char or varchar data type. Similarly, CHAR data type should be used if the value will be of fixed length and VARCHAR if the value is likely to be of variable length. All these decisions have an impact on the ease of coding, performance and storage. However, it may be noted that for the same data element, these decisions would vary from application to application based on the nature of usage of the data element.

Also, as much as possible, ANSI SQL Data Types should be used to store data; however, many RDBMS provide special data types that are designed for efficient storage of these special data types. These data types may be used only if there is no intention to migrate the database to another RDBMS. Also, as much as possible, constraints should be imposed at the database level so that it can be ensured that any backend tinkering will not violate the constraints and thus ensure the data quality.

2.1.2.2 Data Length

Data size should be analysed before deciding on the data type as this will determine what data type is most appropriate for the data element. Length is the maximum number of char data element or the maximum number of digits and numbers of decimal positions for numeric data elements. Depending on data element, the length should be defined; a default value of 255 characters should not be taken.

For example, full name may have 50 characters; name of office may be of 75 characters instead of 255 characters by default. For short names, it can be 25 characters and for acronyms it can be 10-15 characters. Fixing of length helps in better user interface whether on screen display or reports in addition to truncations while transfer of data to information systems.

2.1.3 DATA DOMAIN

2.1.3.1 Acceptable Values

A data element may have a pre-defined set of values which it can take from specification of its domain; these values either can be specifically listed or referenced in a table, or selected from a specified range of values.

One has to be careful while giving list of acceptable values. The options should also be there, in case the set of value is not exhaustive like option for Others, Not Applicable, Not Available etc.

2.1.3.2 Default Values

The value for the data element if a value otherwise is not entered for it, should be available. This helps in case of data migration. A lot of system errors are generated at the back-end as exceptions due to non-checking of null value. Instead of using Null value allowed, it is better to give a default value or Not Available option.

2.1.3.3 Mandatory or Optional

It should be indicated if a value for the data element is optional.

2.1.4 VALIDATIONS

Data input validation is critical to ensuring data quality. It serves the following purposes:

- Prevent malformed, junk, syntax-wise wrong data from entering the database
- Ensures consistency, validity, authentication with masters and external verifiers
- Ensures data is meaningful and conforms to the business rules and constraints
- Prevent unauthorized access and attack to enhance application security

Validations may include data type validations (allowing only numbers in a numeric data element), allowed values validations, range validations, validation against regular expressions such as email etc. These are common validations which need to be identified for every data element. In addition, there may be domain-specific business rules which are applicable on the data element. These rules should also be validated before the data is accepted into the system. Any application can ensure a fairly strong set of validations by adopting the following options:

- Use of drop-down list option, radio button etc to take input from a fixed set of values for which masters are available locally or in an external repository/service
- Use regular expressions to validate structured data like Date fields, Email Id, Mobile No., PIN Code, Aadhaar No., PAN No., etc.
- Prevent use of non-printable special characters (except space, tab, CR, LF) in text boxes
- Define maximum and minimum field length for text fields.
- Define maximum and minimum values for numeric and date fields.

All validation checks need to be enforced at client side as well as server side. It is a common practice among developers to assume that no validation is required at server end if a drop-down list of values is given for the user to select from; this is an incorrect practice as the data can be manipulated over the network.

2.1.5 VERIFICATION

We need to see whether the data is verifiable or not? If it is verifiable, what mechanism is to be followed to get it verified? A sample set of verifications is given in the Appendix on Sample Data Verifications.

For example, email address is a verifiable data element where in verification link can be sent to email address and on receiving the request from verification link, the email address is understood to have been verified. Similar way sending an OTP to mobile number or getting a missed call from a phone number can verify mobile/phone number.

2.1.6 DATA AVAILABILITY

Data availability concerns primarily with master tables and code directories. Some of them such as State, District etc. are generic and some code directories may be specific to certain domain. In order to ensure interoperability among e-Governance applications, it is imperative that data from the source Code Directory is used by all e-Governance applications. The appropriate code directory should be identified at design time itself so that data type etc is in alignment with what is defined in the code directory. The data available in code directories may be accessed through web services from the source system (if available) or a local copy kept with mechanism to sync the local copy with the source system.

Three main code directories have been identified so far. More directories would be added as and when they become available. The three code directories that are currently available include:

1. Local Government Directory (<http://lgdirectory.gov.in>) which maintains the list of land regions (state, district, sub-districts, village) and urban and rural local governments. Data for these data elements should be taken from LGD.
2. data.Gov – Some directories such as PINcode are available in data.gov.
3. Controlled Vocabulary Services (<http://vocab.nic.in>) is another vocabulary service from where data such as list of Ministries, Departments etc. can be taken

2.1.7 USER INTERFACE

Another important aspect is what kind of interface should be provided to user for a specific data item.

2.1.7.1 Data Input

Every data element should be analysed to determine the appropriate input control to be used for capturing the data. Selection of an appropriate control is necessary from the point of view of user friendliness, aesthetics and ensuring validations.

For example, instead of using a text box to capture a date field and apply the whole gamut of validations to ensure that a valid date is captured, it would be more appropriate to use a Date Picker control. Similarly, giving a radio button with just one choice can force the user to abandon the entered data if he/she selects the radio button by mistake; a more appropriate control would be a check box.

As mentioned earlier, every data element should be analysed for the list of allowed values. If such a list is available, then appropriate control such as radio button, check box, combo or list box should be selected as an input mechanism for data entry. This will reduce the number of client-side validations that have to be implemented to restrict the user from entering values outside the list of values. In cases where the allowed values have a specific format such as email, regular expressions may be implemented which is a more elegant way of restricting the user to enter only values in specified formats.

2.1.7.2 Generic Interface Guidelines

This section gives some generic guidelines to be followed while deciding the data type to be used for storing in the database and the type of control that should be preferably used for any data element.

In case, there are fixed number of value that can be chosen, a combo box can be a better option. As such, following can be the general rules.

S. No.	Type of Data Item	Size Limit	Interface Options
1	Character or Varchar	50	Text Box
2	Varchar	50-75	Search as type on with Advanced Search option
3	Varchar	> 50	Text Box or Text area
4	Char Codes with List of Values	4	Radio Button or Combo Box
5	Char Codes with List of Values	15	Combo Box
6	Char Codes with List of Values	> 15 and < 75	Combo Box
7	Char Codes with List of Values	> 75	First 15 more frequently used And with option for more in Combo Click on More should open up Pop-up
8	Date		Date Picker

2.1.7.3 Input using List of Values

This list of values should be in a certain order, by default it should alphabetically. However in cases where there is a sorting order depending on the data element domain for example hierarchical level, it should in that order only.

2.1.7.4 Input using Search

There can be multiple options for providing help during data input to maintain the data quality. Some of options are briefly explained below which can be used independently (say auto-complete only) or in combination (say Text Search along with Alphabetical list).

- **Auto Complete:** Auto complete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values. The auto complete attribute works with the following <input> types: text, search, url, tel, email, password, date pickers, range, and color.
- **Find as you type:** “Find as you type” is a feature that shows quick values as we type in text box or drop-down list.
- **Navigational:** Navigation is based on information stored in hierarchy (parent-child relationship is available) e.g. on selection of Office, system displays list of sub-offices and so on.
- **Text Search:** The value is entered in text box to search for similar/exact names from database
- **Alphabetic List:** The value is selected from a list provided alphabetically from element domain.
- **Most Frequently Used:** The value is selected from an auto-list maintained for specific set of users. For example, set of designations used in a specific office, should be shown at first instance.

2.1.7.5 Field Caption

The field caption is the default display screen prompts or report column heading when the information system outputs the data elements. For example, for group of data element day_birth, month_birth, and year_birth, the caption may be Date of Birth.

2.1.7.6 Output Format

The arrangement of the data element when users see it printed in reports or displayed on the screen should be documented as the length used on a screen or printed lengths may differ.

For example the date should be displayed as dd/mm/yyyy where as it may be printed as

dd-Month-Year. In case of identifiers, for example Credit Card Number is shown in four groups of 4 digits each.

2.1.8 METADATA STANDARDS

This subsection documents the standards that need to be used with respect to various data sets.

Before going ahead with any development, one needs to check for availability of standards related to the data item to know for its data type, size etc. If a Standard for a data element is available, attempt should be made to adhere to the standards to the full so that interoperability with other systems is ensured. In case any issue with respect to the definition in the standard is identified, the same should be brought to the notice of the Standards Committee so that changes to the Standard could be effected.

Some standards have been published in Standards for e-Governance Applications (<http://egovstandards.gov.in>). For example, MDDS – Demographic (Person Identification and Land Region Codification) Standard includes standards for generic data elements which are

- Common across all the domains
- Specific to Person Identification
- Specific to Land Region Codification

On the similar lines specifications need to be prepared for other domains as well. Before going ahead in any development, one needs to check for availability of standards related to the data item to know for its data type, size.

For example, e-Governance Standards need to be followed for person name, gender, date of birth, etc.; a list of sample data item is given in Appendix for Sample Data Item Standards.

2.2

Record Element

This section deliberates on group of related data elements termed as record i.e. a record is a meaningful combination of related data elements that is included in a data flow or retained in a data store.

2.2.1 RECORD IDENTIFICATION

Every record should have information about

- A record identifier, i.e. primary key which is internal to system.

- An additional comment or remarks area
- Created by whom
- Created on
- Last Updated by
- Last updated on
- Verified by, if required
- Verified on, if required

2.2.2 RECORD LEVEL VALIDATION

Data is validated from more than one field with reference to each other so as to ensure correct data input. An example in an 'employee details' database would be where the retirement date of an employee cannot be the same as the start date, or in another database where a value in a field is always greater than another in the same record.

We should ensure record level validation to:

- limit or control the type of data a user can enter in a database or record
- compare the values of two or more fields in the same database or record to ensure they comply with the rules for the database

For example, in scholarship portal, it is mandatory have parent/guardian name, the validation will be all the three names i.e. Father Name, Mother Name and Guardian Name cannot be blank, at least one of them should have a value.

2.3

Data Functions

This section deliberates on issues related to data stores generally referred as tables or entities or data functions. A data function represents functionality provided to the user to meet internal and external data storage requirement. It is a user recognizable group of logically related data or control information maintained in the form of database tables.

2.3.1 DATA FUNCTION/TABLE IDENTIFICATION

2.3.1.1 Table Name

Name of the data function should be meaningful to users and clear at the database level. The name of the table/function should be unique in the data dictionary to avoid any ambiguity. It is better to avoid abbreviation in the table names as far as possible. For example, use Scholarship_Category instead of Scho_Cat or simple Category as table name to store list of scholarship categories.

A naming convention should be adopted as explained above for data elements.

2.3.1.2 Primary Key for a Table

Every table may have a primary key which should be independent of any attributes. The data element name may be <Table_Name>_Id, in case it is integer or <Table_Name>_Code if it is a code generated during the insert record process and is alphanumeric in nature.

2.3.2 REFERENTIAL INTEGRITY

Referential integrity should be maintained within the database for all applicable data elements.

2.4

Identifiers

In this section, pros and cons of some of the identifiers are discussed along with criteria for defining identifiers.

2.4.1 CRITERIA FOR DEFINING OF NEW IDENTIFIER

The identifiers should satisfy the following criteria which have been explained in paper on UID Numbering Scheme.

- Availability
 - o Format, Structure, Length should be enough to take care of future requirements
- Longevity
- Privacy
- Numeric Value to cater to multi-lingual society
- Simple to remember
- Activation/De-Activation Process
- Recovery Process
- Reserved Numbers, if any
- 1st digit should not be zero
- Provision for check digit
- Semantics-free

2.4.2 COMMON IDENTIFIERS

2.4.2.1 Aadhaar

Aadhaar number is a 12-digit random number issued by the UIDAI (“Authority”) to

the residents of India after satisfying the verification process laid down by the Authority. Aadhaar number is verifiable in an online, cost-effective way. It is unique and robust enough to eliminate duplicates and fake identities and may be used as a basis/primary identifier to roll out several Government welfare schemes and programmes for effective service delivery thereby promoting transparency and good governance.

The Aadhaar structure is as follows: 9999 9999 9999:

- All 12 characters are digits
- The first eleven digits is a random number
- The last character is a check digit calculated using Verhoeff algorithm.

2.4.2.2 Permanent Account Number (PAN)

Permanent Account Number (PAN) is a code that acts as identification for individuals and entities that may be covered under Income Tax act. It is a unique, 10-character alphanumeric identifier, issued by the Indian Income Tax Department. An example number would be in the form of ABNPA0061H.

The PAN is unique to each individual and is valid for the life time of the holder, throughout India. An important point to note would be that once issued, the PAN is not affected by a change of address.

The PAN structure is as follows: AAAPL1234C:

- First five characters are letters, next four numerals, last character letter.
- The first three letters are sequence of alphabets from AAA to ZZZ
- The fourth character informs about the type of holder of the card.
- The fifth character of the PAN is the first character
 - o of the surname or last name of the person, in the case of a “Personal” PAN card, where the fourth character is “P” or
 - o of the name of the Entity, Trust, society, or organisation in the case of Company/ HUF/ Firm/ AOP/ BOI/ Local Authority/Artificial Judicial Person/Govt., where the fourth character is “C”, “H”, “F”, “A”, “T”, “B”, “L”, “J”, “G”.
- The last character is an alphabetic check digit.

The pros and cons of PAN are as follows

1. Being multi-lingual society, it can have only English character set.

- a. Because of its alphanumeric structure
 - b. Usage point of view as keyboard has to have English Character Set along with digits and its integration with IVRS.
2. The number has built-in business intelligence i.e. 4th Character identifies it as a Person or Entity and 5th character identifies 1st alphabet of Entity Name or Person last/surname. In case of any change in these, it loses its sanctity or a new number is to be provided.
3. The only plus point with PAN is its institutional framework for providing various services like know your PAN; PAN verification; PAN data change management etc.

2.4.2.3 District Information System for Education (DISE) Code

U-DISE (Unified District Information System for Education) is a database of information about schools in India. The database was developed at the National University for Educational Planning and Administration. The DISE code comprises of 11 characters (AABBCCDDDEE) and its structure is as follows:

- First two characters (AA) is two digit code for State
- First four characters (AABB) are four digit code for district
- Next seven characters (digits) identifies a school in a district and comprises of
 - o Two characters (CC) is two digit code block within district
 - o Three characters (DDD) is three digit for village in a block/district
 - o Two characters (EE) is two digit for school code in a village

The number has built-in business intelligence i.e. it comprise of State, District, Village codes. In case of any change in these, it loses its sanctity or a new number is to be provided for the same school. There is no check digit.

2.4.2.4 Labour Identification Number (LIN)

LIN is a 10-digit random number issued by Ministry of Labour & Employment to establishment covered under any of the labour laws. The objective is to move towards LIN only regime by replacing all other registration numbers issued under various Labour Acts by EPFO, ESIC, CLC(C), DGMS and State Labour Depts. With the integration of LIN establishments will be able to file monthly/annual returns, enforcement agencies using LIN only and carry out inspections with transparency and good governance.

The services for LIN Verification have been developed and integrated with EPFO and ESIC system.

The LIN structure is as follows: V-9999-9999-C:

- All 10 characters are digits
- The first character is reserved for versioning with 9 reserved for foreign entities
- Next Eight digits is a random number
- The last character (10th) is a check digit based on Verhoeff Algorithm.

2.4.2.5 Indian Financial System Code (IFSC)

The Indian Financial System Code (IFS Code or IFSC) is an alphanumeric code that facilitates electronic funds transfer and uniquely identifies each bank branch. The IFSC is an 11-character code with the first four alphabetic characters representing the bank name, and the last six characters (usually numeric, but can be alphabetic) representing the branch. The fifth character is 0 (zero) and reserved for future use.

2.5

Guidelines for Common Data Elements

This guidelines for most commonly used data elements in terms of the data quality parameters are detailed in appendices. For specification of data common data elements, Snake Case (Lower Case Embedded Underscore) has been chosen as naming convention.

CH 03

AUTHENTICATION

- 3.1 AUTHENTICATION LEVELS**
 - 3.1.1 SINGLE-FACTOR AUTHENTICATION**
 - 3.1.2 TWO-FACTOR AUTHENTICATION [2FA]**
 - 3.1.3 MULTI-FACTOR AUTHENTICATION**
- 3.2 AUTHENTICATION TYPES**
 - 3.2.1 HTTP BASIC AUTHENTICATION**
 - 3.2.2 FORM BASED AUTHENTICATION**
 - 3.2.3 DIGITAL CERTIFICATES (SSL AND TLS)**
 - 3.2.4 ONE TIME PASSWORD:**
 - 3.2.5 BIOMETRIC AUTHENTICATION**
- 3.3 IMPLEMENTATION OF AUTHENTICATION**
 - 3.3.1 HTTP BASIC/ FORM BASED AUTHENTICATION**
 - 3.3.2 AUTHENTICATION USING DATABASE**
 - 3.3.3 AUTHENTICATION USING LDAP**
 - 3.3.4 CERTIFICATE BASED AUTHENTICATION**
 - 3.3.5 ONE TIME PASSWORD BASED AUTHENTICATION**
 - 3.3.6 CUSTOM APPLICATION GENERATED OTP**
 - 3.3.7 AADHAAR BASED OTP.**
 - 3.3.8 TIME BASED ONE TIME PASSWORD (TOTP)**
 - 3.3.9 BIOMETRIC BASED AUTHENTICATION**
- 3.4 SIGN-UP/LOGIN PROCESSES**
 - 3.4.1 WAYS TO SIGN-UP**
 - 3.4.1.1 SIGN-UP USING APPLICATION SPECIFIC USER-ID/PASSWORD**
 - 3.4.1.1.1 ON-LINE SIGN-UP**
 - 3.4.1.1.1.1 CONFIRM EMAIL FOR SIGN-UP**
 - 3.4.1.1.1.2 SET USER-ID/PASSWORD (CONFIRM MOBILE)**
 - 3.4.1.1.2 OFF-LINE SIGN-UP**
 - 3.4.1.1.3 BULK SIGN-UP**
 - 3.4.1.1.3.1 BULK SIGN-UP USING NAME, ADDRESS**

3.4.1.1.3.2	BULK SIGN-UP USING EMAIL AND MOBILE NUMBER
3.4.1.2	SIGN-UP USING OFFICIAL USER-ID/PASSWORD (E.G. NIC EMAIL)
3.4.1.3	SIGN-UP USING SOCIAL NETWORKING USER-ID/PASSWORD
3.4.2	WAYS TO RECALL SIGN-UP CREDENTIALS
3.4.2.1	USERS WITH EMAIL/MOBILE
3.4.2.1.1	FORGET USER-ID
3.4.2.1.1.1	KNOW USER-ID
3.4.2.1.2	FORGET PASSWORD
3.4.2.1.2.1	SET PASSWORD
3.4.2.2	USERS WITHOUT EMAIL/MOBILE
3.4.2.2.1	RESET USER-ID/PASSWORD
3.4.3	WAYS TO LOGIN
3.4.4	WAYS TO CHANGE/DEACTIVATE CREDENTIALS
3.4.4.1	CHANGE PASSWORD
3.4.4.2	SUBMIT EMAIL DE-ACTIVATION REQUEST
3.4.4.2.1	DE-ACTIVATE EMAIL ADDRESS
3.4.4.3	SUBMIT MOBILE DE-ACTIVATION REQUEST
3.4.4.4	DEACTIVATE USER-ID
3.5	ADDITIONAL BEST PRACTICES
3.5.1	STOP AUTO USER CREATION
3.5.2	USING CAPTCHA
3.5.3	CONTEXT BASED AUTHENTICATION
3.5.4	ADDITIONAL IMAGE BASED PROFILE VERIFICATION
3.5.5	USING FORGOT PASSWORD
3.5.5.1	RESET PASSWORD LINK
3.5.5.2	TEMPORARY PASSWORD
3.5.6	USING PROFILE/TRANSACTIONAL PASSWORD
3.5.7	SECURITY QUESTIONS
3.5.8	NEW ACCOUNT ACTIVATION LINKS
3.5.9	ACCOUNT LOCKING
3.5.10	ACCOUNT AUDIT POLICY
3.6	OTHER GUIDELINES & CONCLUSION

AUTHENTICATION

3.1

Authentication Levels

The ways in which someone may be authenticated fall into three categories, based on what are known as the factors of authentication:

- Something the user knows,
- Something the user has, and
- Something the user is.

Each authentication factor covers a range of elements used to authenticate or verify a person's identity prior to being granted access, approving a transaction request, signing a document or other work product, granting authority to others, and establishing a chain of authority.

It is suggested that for positive authentication, elements from at least two, and preferably all three, factors should be verified. The three factors (classes) and some of elements of each factor are:

- the knowledge factors: Something the user knows (e.g., a password, Partial Password, pass phrase, or personal identification number (PIN), challenge response (the user must answer a question, or pattern), Security question etc.
- the ownership factors: Something the user has (e.g., wrist band, ID card, security token, cell phone with built-in hardware token, software token, or cell phone holding a software token)
- the inherence factors: Something the user is or does (e.g., fingerprint, retinal pattern, DNA sequence (there are assorted definitions of what is sufficient), signature, face, voice, unique bio-electric signals, or other biometric identifier).

In a web application it is easy to confuse authentication and session management (dealt with in a later section). Users are typically authenticated by a username and password or similar mechanism. When authenticated, a session token is usually placed into the user's browser (stored in a cookie). This allows the browser to send a token each time a request is being made, thus performing entity authentication on the browser. The act of user authentication usually takes place only once per session, but entity authentication takes place with every request.

3.1.1 SINGLE-FACTOR AUTHENTICATION

As the weakest level of authentication, only a single component from one of the three categories of factors is used to authenticate an individual's identity. The use of only one factor does not offer much protection from misuse or malicious intrusion. This type of authentication is not recommended for financial or personally relevant transactions that warrant a higher level of security.

3.1.2 TWO-FACTOR AUTHENTICATION [2FA]

When elements representing two factors are required for authentication, the term two-factor authentication is applied — e.g. a bankcard (something the user has) and a PIN (something the user knows). Business networks may require users to provide a password (knowledge factor) and a pseudorandom number from a security token (ownership factor). Access to a very-high-security system might require a mantrap screening of height, weight, facial, and fingerprint checks (several inherence factor elements) plus a PIN and a day code (knowledge factor elements), but this is still a two-factor authentication.

3.1.3 MULTI-FACTOR AUTHENTICATION

Instead of using two factors as used in 2FA, multiple authentication factors are used to further enhance security of a transaction in comparison to the 2FA authentication process.

3.2

Authentication Types

The authentication type in web applications can be based on the sensitivity of the application. Single level or multiple level authentication of authentication type mentioned above can be made available in the application. Various guidelines are given for the authentication types and levels in the following section of this document.

3.2.1 HTTP BASIC AUTHENTICATION

HTTP Basic authentication is a method for the client to provide a username and a password when making a request. This is the simplest possible way to enforce access control as it doesn't require cookies, sessions or anything else. Username and password can be stored in databases or any user directories as per the need of the organization.

3.2.2 FORM BASED AUTHENTICATION

Rather than relying on authentication at the protocol level, web based applications can use code embedded in the web pages themselves. Specifically, developers have previously used HTML FORMs to request the authentication credentials (this is supported by the TYPE=PASSWORD input element). This allows a designer to present the request for

credentials (Username and Password) as a normal part of the application and with all the HTML capabilities for internationalization and accessibility.

3.2.3 DIGITAL CERTIFICATES (SSL AND TLS)

Both SSL and TLS can provide client, server and mutual entity authentication. Digital certificates are a mechanism to authenticate the providing system and also provide a mechanism for distributing public keys for use in cryptographic exchanges (including user authentication if necessary). The most common usage for digital certificates on web systems is for entity authentication when attempting to connect to a secure web site (SSL). Most web sites work purely on the premise of server side authentication even though client side authentication is available. This is due to the scarcity of client side certificates and in the current web deployment model this relies on users to obtain their own personal certificates from a trusted vendor; and this hasn't really happened on any kind of large scale.

For high security systems, client side authentication is a must and as such a certificate issuance scheme (PKI) might need to be deployed. Further, if individual user level authentication is required, then 2-factor authentication will be necessary.

3.2.4 ONE TIME PASSWORD:

A one-time password (OTP) is a password that is valid for only one login session or transaction, on a computer system or other digital device. OTPs avoid a number of shortcomings that are associated with traditional (static) password-based authentication; a number of implementations also incorporate two factor authentication by ensuring that the one-time password requires access to something a person has (such as a cellphone) as well as something a person knows (such as a PIN).

3.2.5 BIOMETRIC AUTHENTICATION

Biometric authentication is a security process that relies on the unique biological characteristics of an individual to verify that he is who he is. Biometric authentication systems compare a biometric data capture to stored, confirmed authentic data in a database. If both samples of the biometric data match, authentication is confirmed.

3.3

Implementation of Authentication

3.3.1 HTTP BASIC/ FORM BASED AUTHENTICATION

This is the most commonly used types of authentication in any web application. In this type of username and password are used. User base can be stored either in Database or any user directory such as LDAP or Active Directory.

3.3.2 AUTHENTICATION USING DATABASE

Database can authenticate users attempting to connect to a database, by using information stored in that database itself. To set up Database to use database authentication, create each user with an associated password. The user must provide this user name and password when attempting to establish a connection. This process prevents unauthorized use of the database, because the connection will be denied if the user provides an incorrect password. Database should store user passwords in the database tables in an encrypted format to prevent unauthorized alteration. Users can change their passwords at any time.

Database authentication should include the following features:

- Password Encryption While Authenticating.
- Account Locking
- Password Lifetime and Expiration
- Password History
- Password Complexity Verification

3.3.3 AUTHENTICATION USING LDAP

LDAP (Lightweight Directory Access Protocol) is a software protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet. To access the LDAP service, the LDAP client first must authenticate itself to the service. That is, it must tell the LDAP server who is going to be accessing the data so that the server can decide what the client is allowed to see and do. If the client authenticates successfully to the LDAP server, then when the server subsequently receives a request from the client, it will check whether the client is allowed to perform the request. This process is called access control.

In LDAP, authentication is supplied in the “bind” operation. Ldapv3 supports three types of authentication: anonymous, simple and SASL authentication. A client that sends a LDAP request without doing a “bind” is treated as an anonymous client. Simple authentication consists of sending the LDAP server the fully qualified DN of the client (user) and the client’s clear-text password. This mechanism has security problems because the password can be read from the network. To avoid exposing the password in this way, you can use the simple authentication mechanism within an encrypted channel (such as SSL), provided that this is supported by the LDAP server.

Finally, SASL is the Simple Authentication and Security Layer (RFC 2222). It specifies a challenge-response protocol in which data is exchanged between the client and the server for the purposes of authentication and establishment of a security layer on which to carry out

subsequent communication. By using SASL, LDAP can support any type of authentication agreed upon by the LDAP client and server.

Various guidelines are given for such type of authentication.

1. Password should not be stored in plain text in database or any user directory
2. Password should be stored preferably with salted SHA2 encrypted values
3. Password should not be same as username
4. Password should not travel as plain text on network
5. Password should be minimum 8 Characters with at least 1 Capital, 1 small, 1 special character and 1 number
6. Username and password request should be sent using POST method.
7. Password should not be in a pattern eg. Abcd@123
8. Proper session time out should be maintained
9. Proper logout of session should be in application even if user closes the application without logout from application

[Refer NIC Password Policy <https://security.nic.in/docs/password%20policy.pdf>]

3.3.4 CERTIFICATE BASED AUTHENTICATION

This method ensures authentication using a public and private encryption key that is unique to the authentication device and the person who possesses it. The methodology is to first register the DSC from the user and keep their Public key along with Serial No. and Name etc. Later when user selects the same DSC at login page then encrypt some data from their private key and transfer the same over internet to Web application and decrypt using their registered public key (identifying the public key using serial no and name). The PC can then be authorized for certain period using cookies.

Method of DSC based authentication

1. Login seed is set as any random number generated; done on server side
2. Sign Login seed with private key in applet
3. Get original value of signed seed on server side
4. Verify login seed with signed seed to authenticate on server seed.

3.3.5 ONE TIME PASSWORD BASED AUTHENTICATION

This type of authentication is generally used as secondary authentication and adds as an extra layer of security and personalization in the application. Web application can use One Time Password (OTP) facilities in the following format

1. Custom Application generated OTP
2. Aadhaar based OTP
3. Time Based OTP

3.3.6 CUSTOM APPLICATION GENERATED OTP

Custom based OTP is a random number or a salt can be generated from the application and same can be verified by the user input. In this type SMS and Email gateway integration is required with the application. User Mobile and Email id should also a mandatory requirement for this type of authentication.

Guidelines for Custom Based OTP

1. Make Mobile number and Email id in user profile mandatory.
2. Mandatory checks needs to be put up on changing mobile number and email id in profile section. Intimation on both old and new mobile number and email id should be given of any change in it.
3. OTP should be generated using any standardized algorithm and end result should be stored in encrypted format so that nobody can easily identify the OTP
4. OTP input should also be encrypted to travel in network.
5. Resending of OTP should also be provided.

3.3.7 AADHAAR BASED OTP.

Authentication with Aadhaar based OTP is provided by UIDAI. In this authentication Aadhaar Number needs to be submitted or picked from user profile and authentication is reduced to a 1:1 match. An OTP is sent to the registered mobile number with UIDAI and on user input Aadhaar authentication service only responds with a “yes/no”.

Guidelines for Aadhaar Based OTP Authentication

1. Aadhaar number should be made mandatory in user profile, if this type of authentication is used.
2. Aadhaar number field should be exactly 12 digits long and it should only contain numeric numbers.

3. Provision of seeding/update in Aadhaar number should be done with the use of OTP service only so that correct Aadhaar number is seeded for said user.
4. The mobile number should be registered with Aadhaar

3.3.8 TIME BASED ONE TIME PASSWORD (TOTP)

The Time-based One-time Password Algorithm (TOTP) is an algorithm that computes a one-time password from a shared secret key and the current time. It has been adopted as Internet Engineering Task Force standard RFC 6238, is the cornerstone of Initiative for Open Authentication (OATH), and is used in a number of two-factor authentication systems.

TOTP is an example of a hash-based message authentication code (HMAC). It combines a secret key with the current timestamp using a cryptographic hash function to generate a one-time password. The timestamp typically increases in 30-second intervals, so passwords generated close together in time from the same secret key will be equal.

Applications of TOTP

- Banking/Finance
- Healthcare
- Public Sector
- Homeland Security
- Professional Services
- Corporate Security
- Cloud Computing Security
- Online Banking Security
- Mobile Banking Security
- E-Commerce Security
- VPN Access Security
- Network Access Security
- Identity Management
- Embedded Token
- Mobile Authentication
- Software-as-a-Service (SaaS)

Strengths

- Easy to setup (using QR code scanning)
- The TOTP passwords are short-lived; they only apply for a given amount of human time.

- Network availability is not required.
- No need of a separate device like USB tokens etc.
- Light-weight
- Man-in-middle attack can be avoided
- Can be easily integrated
- Can be used along with other modes of ota (over the air) techniques of otp provisioning like sms, Bluetooth, wap etc.

3.3.9 BIOMETRIC BASED AUTHENTICATION

Biometric Based authentication can be used where personal and physical identity of the user needs to be captured before authenticated in the system. Biometrics are captured using biometric devices and the input is passed to be matched with the details stored in the user profile. Aadhaar based biometric service is useful in this kind of authentication and storing biometrics is eliminated and only integration of authentication service is required.

Guidelines for Aadhaar Based Biometric Authentication

1. Aadhaar number should be made mandatory in user profile, if this type of authentication is used.
2. Aadhaar number field should be exactly 12 digits long and it should only contain numeric numbers.
3. Provision of seeding/update in Aadhaar number should be done with the use of OTP service only so that correct Aadhaar number is seeded for said user.
4. Common API for Biometric capture should be considered unless the application needs to be bound to any specific device.

3.4

Sign-up/login Processes

Before we go ahead, it need to be ensured that one should user “Sign-up” rather “Register” for providing a user-id/password as the term “Register” has a different meaning in legal terms. Now, a typical sign-up form contains a couple of fields, the objective is to identify a person to whom we are going to provide access for using the application and facilitate with single sign-on.

3.4.1 WAYS TO SIGN-UP

3.4.1.1 Sign-up using Application Specific User-id/Password

3.4.1.1.1 On-line Sign-up

The sign-up form may ask for minimum information (like name, email, mobile, verification/captcha code) with mobile number verification using OTP and email verification using activation hyperlink.

3.4.1.1.1.1 Confirm Email for Sign-up

The web application will prompt for verification code (like captcha code) on receiving the activation hyperlink sent through email during signup.

3.4.1.1.1.2 Set User-id/Password (Confirm Mobile)

After verification of email address and successful entry of verification code, the system will prompt for OTP sent (with resend option) on mobile, desired user-id (with facility to check for availability) and password to login.

3.4.1.1.2 Off-line Sign-up

The option is to be provided in case of non-availability of email address and mobile number. The sign-up form may ask for minimum information (like name, address, date of birth, temporary user-id, and password) from the user who has been authorised to create user-id/password based on physical information available. Once the user-id/password is created, the same has to be sent to the user at his/her mailing address. On first login, the user must change user-id/password.

3.4.1.1.3 Bulk Sign-up

The option is to be provided for creation of bulk users by Nodal Officer who has been authorised to create user-id/password based on physical information available. There can be two type of sign-up information.

3.4.1.1.3.1 Bulk Sign-up using Name, Address

The bulk user form may ask for minimum information (like name, address, date of birth, temporary user-id, and password) from Nodal Officer. Once the user-id/password is created, the same has to be sent to the respective users at their mailing address. On first login, the user must change user-id/password to keep it personalised.

3.4.1.1.3.2 Bulk Sign-up using Email and Mobile number

The bulk user form may ask for minimum information (like name, email address, mobile number) from Nodal Officer. Once the database of temporary user-id/password is created, intimation (activation link) has to be sent to the respective users at their email address for setting as their user-id/password to keep it personalised. This will ensure correctness of email address and mobile numbers along with personalisation using Confirm Email and Set User-id/Password as explained above.

3.4.1.2 Sign-up using Official User-id/Password (e.g. NIC Email)

In order to facilitate the official users with single sign-on to use their existing user-id/password, the sign up form may ask for Authentication Server related information (necessary interfaces need to be provided), user-id and password and may follow the process accordingly.

3.4.1.3 Sign-up using Social Networking User-id/Password

In order to facilitate users with single sign-on to use their existing social networking user-id/password, the sign up form may ask for relevant information (necessary interfaces need to be provided) and may follow the process accordingly.

3.4.2 WAYS TO RECALL SIGN-UP CREDENTIALS

3.4.2.1 Users with Email/Mobile

3.4.2.1.1 Forget User-id

The forget User-id form may ask verification code (like captcha), registered/verified email-id and mobile, a message/hyperlink will be sent on email and OTP on mobile.

3.4.2.1.1.1 Know User-id

The web application will prompt for verification code on receiving the Forget User-id hyperlink sent in registered/verified email. The system will ask for OTP sent on registered/verified mobile and will display the user-id.

3.4.2.1.2 Forget password

The forget password form may ask verification code (like captcha), user-id and will confirm the email-id and mobile (both in masked mode), an OTP will be sent on registered/verified email and mobile.

3.4.2.1.2.1 Set Password

The web application will prompt for OTP sent for setting the password and new password. Alternatively, Nodal Officer (User Administrator) may be approached for resetting of password.

3.4.2.2 Users without Email/Mobile

3.4.2.2.1 Reset User-id/Password

The Reset User-id/Password form may ask Nodal Officer for name, date of birth, and other such relevant information to uniquely identify the user based on physical request. The Nodal Officer can reset the password. The user-id/password has to be sent to mailing address or has to be handed over in person to respective user in sealed envelope. On first

login after reset, the user must use Change Password option to change the password and keep it personalised.

3.4.3 WAYS TO LOGIN

In order to avoid confusion between sign-in/on and sign-up, it is suggested to use term “Login” instead of term “Sign-in”. The login form may ask for verification/captcha code should facilitate the user to login using multiple options like

1. Application Specific User-id/Password
2. OTP based Mobile number or Aadhaar number
3. Official User-id (like NIC Email-id/password)
4. Google/Facebook/other social networking user-id/password for public

3.4.4 Ways to Change/Deactivate Credentials

3.4.4.1 Change Password

The Change Password form may ask old password, new password (along with re-enter new password) to logged-in users for changing their password only. The intimation need to be sent to registered email/mobile.

3.4.4.2 Submit Email De-activation Request

In case of wrong entry of email address, an option is required to de-activate email address. The Email De-Activation Form may ask for email address and verification/captcha code to delete wrong email address registered in the portal. On submission of request, a confirmation email will be sent along with email de-activation hyperlink.

3.4.4.2.1 De-activate Email Address

The web application will prompt for verification code (like captcha code) on receiving the Email De-activation hyperlink sent through email. On successful entry of verification code, the system de-activates/blocks the email address from the database. Once email has been deactivated, the credential information, activation hyperlinks, OTPs will not be sent to the de-activated email address. It can be re-activated only by requesting Nodal Officer/User Administrator.

3.4.4.3 Submit Mobile De-activation Request

In case of wrong entry of mobile number, an option is required to de-activate mobile number. The mobile de-activation Form may ask for mobile number, verification/captcha code followed by OTP sent on the mobile. On successful entry of OTP, the mobile number will be de-activated/blocked in the database. Once mobile number has been deactivated,

the credential information, OTPs etc will not be sent to the de-activated mobile. It can be re-activated only by requesting Nodal Officer/User Administrator.

3.4.4.4 Deactivate User-id

The user may choose the option Change Password, email/mobile deactivation options to deactivate the user-id.

3.5

Additional Best Practices

3.5.1 STOP AUTO USER CREATION

Lock down critical weak points such as the ability to create new user accounts with system admin privileges and other common exploits to prevent catastrophic penetration of infrastructure by attackers.

3.5.2 USING CAPTCHA

A CAPTCHA (a acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart") is a type of challenge-response test used in computing to determine whether or not the user is human.

CAPTCHAs can be used to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring it to solve a CAPTCHA after a certain number of unsuccessful logins. This is better than the classic approach of locking an account after a sequence of unsuccessful logins, since doing so allows an attacker to lock accounts at will. The following guidelines are strongly recommended for any CAPTCHA code:

- **Accessibility.** CAPTCHAs must be accessible. CAPTCHAs based solely on reading text — or other visual-perception tasks — prevents visually impaired users from accessing the protected resource. Any implementation of a CAPTCHA should allow blind users to get around the barrier, for example, by permitting users to opt for an audio or sound CAPTCHA.
- **Image Security.** CAPTCHA images of text should be distorted randomly before being presented to the user. Many implementations of CAPTCHAs use undistorted text, or text with only minor distortions. These implementations are vulnerable to simple automated attacks.
- **Script Security.** Building a secure CAPTCHA code is not easy. In addition to making the images unreadable by computers, the system should ensure that there are no easy ways around it at the script level. Common examples of insecurities in this respect

include: (1) Systems that pass the answer to the CAPTCHA in plain text as part of the web form. (2) Systems where a solution to the same CAPTCHA can be used multiple times (this makes the CAPTCHA vulnerable to so-called “replay attacks”). Most CAPTCHA scripts found freely on the Web are vulnerable to these types of attacks.

- **Security Even After Wide-Spread Adoption.** There are various “CAPTCHAs” that would be insecure if a significant number of sites started using them. An example of such a puzzle is asking text-based questions, such as a mathematical question (“what is 1+1”). Since a parser could easily be written that would allow bots to bypass this test, such “CAPTCHAs” rely on the fact that few sites use them, and thus that a bot author has no incentive to program their bot to solve that challenge. True CAPTCHAs should be secure even after a significant number of websites adopt them.

3.5.3 CONTEXT BASED AUTHENTICATION

Context-based authentication uses contextual information to ascertain whether a user’s identity is authentic or not. Based on risk profiles, organizations may limit access to specific systems or content items based on a user’s criteria, including whether the user is authenticating from the local or remote network, whether the user is accessing information from a corporate computer, or whether the access time appears reasonable (i.e. office hours for that user’s country location based on their computer IP address). Since much of the information used by context-based information is publicly available and therefore easily accessible to hackers, it is recommended to use this method to complement other stronger authentication methods or as a first level of authentication in a multi-layered approach

Guidelines for Context Based Authentication

1. Application can be restricted to Organisational Network if the intended users are only within the organisation.
2. Application can be accessed using VPN if it is required outside the network adding an extra layer of security
3. Application can be restricted in office hours if it is required in office hours only.
4. IP based restriction can be applied on certain administrative accounts.

3.5.4 ADDITIONAL IMAGE BASED PROFILE VERIFICATION

Many banks use image recognition as part of the multi-factor authentication process so their customers can login to their accounts and authorize various financial transactions in a secure environment. On the web, this sort of image recognition authentication is ideal for preventing phishing attacks where another website could mimic the look and feel of your bank. Even if a phishing site looked identical to your bank’s log in page, it would not

know what your authentication image looked like, and would not be able to fool users as easily on that page of the process. Also, that sort of phishing attack would render the bank username and password useless. Even with the password, the data thief would not be able to log in to the account without also knowing the authentication image, which they would not be able to steal in a phishing scheme.

3.5.5 USING FORGOT PASSWORD

Forgot password feature needs to be implemented in such a way that it covers all the possibilities of any attack vector. Initial step during forgot password could be gathering the identity of the user by asking various questions which would in turn prove the identity of the account like, what was the last reset password user remember? Last time user accessed his account? or any other similar questions which reveal the identity of the user.

The next step could be Security Question! Security questions should not be the same old traditional way asking about mother's maiden name/place of birth/ favorite teacher or any such questions which are easily crackable and letting the attacker to easily compromise the account. Instead if the application allows user to set his own set of questions) which could infinite set for an attacker to guess the answer.

One more important thing to remember here is that the input field for accepting the answer for the security question should be of type "password" this would protect from shoulder surfing, most of the applications does not do so, which is of more important. Number of attempts user can try answering the security question should also be limited or CAPTCHA needs to implement to prevent brute force attacks. Once the user provides the correct answer the password can be reset in 2 ways.

- Reset password link with a token associated to it
- Temporary password

One of the worst methods on forgot password feature followed by some developers is directly sending the password which was set by the user during registration. This happens only when the password is saved in clear text. Good practice in storing the password is to be hashed+salted before storing it.

3.5.5.1 Reset password link

- Once reset password link is used, link should be expired for the next use.
- Till the user resets password, the previous password should not be disabled.
- Even if the link is unused the reset password link needs to be expired within a defined time say 48 or 72 hours.
- Reset password link should be over an SSL

- Old/previous password reset link should be expired once new password link is generated.
- Token used in reset password link should be mapped to the users email ID and should not be used to reset password of another user.
- Token should not be sequential or easily guessable or a short one so that it is not easily brute forced.
- Password policy should be maintained on reset password page.
- Display custom error message to avoid username enumeration

3.5.5.2 Temporary Password

- Application should force the user to change the password on the first login.
- If not used the previous password should not be disabled.
- Temporary password should be one time use and should not be able to use it again
- Flush the memory of the browser after password change
- 302 redirection after successfully resetting the password.
- Validity for the temporary password could be not more than 4-5 hours.
- Password should not be the same as old password.

Sending reset password link with a token associated to it is the best and economical way of implementing forgot password feature, it avoids usage of CAPTCHA and increased accessibility.

3.5.6 USING PROFILE/TRANSACTIONAL PASSWORD

Profile or Transactional Passwords are an additional security password that can be implemented over and above the basic authentication. Many bank applications uses profile or transactional password to transact in Accounts related to that profile. This helps to have an extra layer of security.

Profile or Transactional passwords should be created and used same as user passwords. It should always be different from user password and cannot be reset easily.

3.5.7 SECURITY QUESTIONS

Most of us can instantly spot a bad “security question” when we see one. You know the ones we mean. Ones like “What is your favorite color?” are obviously bad. But as the Good Security Questions web site rightly points out, “there really are NO GOOD security questions; only fair or bad questions”.

The reason that most organizations allow users to reset their own forgotten passwords is not because of security, but rather to reduce their own costs by reducing their volume of

calls to their help desks. It's the classic convenience vs. security trade-off, and in this case, convenience (both to the organization in terms of reduced costs and to the user in terms of simpler, self-service) almost always wins out.

So given that the business aspect of lower cost generally wins out, what can we do to at least raise the bar a bit?

Here are some suggestions. Note that we intentionally avoid recommending specific security questions. To do so likely would be counterproductive because many developers would simply use those questions without much thinking and adversaries would immediately start harvesting that data from various social networks.

Any security questions or identity information presented to users to reset forgotten passwords should ideally have the following four characteristics:

- **Memorable:** If users can't remember their answers to their security questions, you have achieved nothing.
- **Consistent:** The user's answers should not change over time. For instance, asking "What is the name of your significant other?" may have a different answer 5 years from now.
- **Nearly universal:** The security questions should apply to a wide an audience of possible.
- **Safe:** The answers to security questions should not be something that is easily guessed, or research (e.g., something that is matter of public record).

3.5.8 NEW ACCOUNT ACTIVATION LINKS

Sending account activation links to registered email and mobile number helps to identify the correct email and mobile number of the user so that any change in profile can be sent to the correct recipient. This helps in alerting the user in case of any attack on its account.

3.5.9 ACCOUNT LOCKING

The most obvious way to block brute-force attacks is to simply lock out accounts after a defined number of incorrect password attempts. Account lockouts can last a specific duration, such as one hour, or the accounts could remain locked until manually unlocked by an administrator.

However, account lockout is not always the best solution, because someone could easily abuse the security measure and lock out hundreds of user accounts. In fact, some Web sites experience so many attacks that they are unable to enforce a lockout policy because they would constantly be unlocking customer accounts. An attacker can cause a denial of service (DoS) by locking out large numbers of accounts.

Account lockout is sometimes effective, but only in controlled environments or in cases where the risk is so great that even continuous DoS attacks are preferable to account

compromise. In most cases, however, account lockout is insufficient for stopping brute-force attacks. Consider, for example, an auction site on which several bidders are fighting over the same item. If the auction Web site enforced account lockouts, one bidder could simply lock the others' accounts in the last minute of the auction, preventing them from submitting any winning bids. An attacker could use the same technique to block critical financial transactions or e-mail communications.

3.5.10 ACCOUNT AUDIT POLICY

Security policy determines whether the system generates audit events when the following user account management tasks are performed:

- A user account is created, changed, deleted, renamed, disabled, enabled, locked out, or unlocked.
- A user account password is set or changed.
- Permissions on accounts that are members of administrators groups are changed.
- A user login details like time, IP and other details
- Logout details
- Failed Login Details

3.6

Other Guidelines & Conclusion

1. Alternative protective mechanism should be created along with any authentication mechanism so that user is not able to login.
2. User auto generation for administrator privileges should be disabled

To sum up we can say the following matrix can be recommended for authentication level and types for various types of Applications

Sectors	Recommended Factors	Username/ Password	DSC	OTP	Biometric	Profile / Transaction Password	CAPTCHA	Security Question	Forgot Password	Image Authentication	Context Based	Account Locking
e-Commerce	2 Level	✓		✓		✓	✓		✓			
G2E (Employee Portals)	2 Level	✓	✓				✓	✓	✓		✓	✓
G2G	1-2 Level	✓	✓				✓				✓	
G2C (Citizen Interfaces)	3 Level	✓						✓	✓			
G2B (eTender, eProcurement)	2 Level	✓	✓	✓	✓		✓	✓	✓		✓	✓
Email Clients	1-2 Level	✓		✓			✓	✓	✓		✓	✓
Wallets/Banking	2 Level	✓		✓		✓	✓	✓	✓	✓		✓
System Administration	2-3 Level	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

 Mandatory
  Optional

4.1	USER-CENTRIC APPROACH
4.1.1	RELATIONSHIP
4.1.2	CONVERSATION
4.1.3	APPEARANCE
4.2	STRUCTURING OF FORM
4.3	FORM ELEMENTS
4.3.1	LABELS
4.3.2	INPUT FIELDS
4.3.2.1	TEXT BOXES
4.3.2.2	RADIO BUTTONS
4.3.2.3	CHECK BOXES
4.3.2.4	DROPDOWN LISTS & COMBO BOXES
4.3.2.5	LIST BOXES
4.3.3	ACTIONS
4.3.4	HELP TEXT
4.3.4.1	FORM LEVEL INSTRUCTIONS
4.3.4.2	INLINE INSTRUCTIONS
4.4	FORM VALIDATIONS
4.4.1	VALIDATION METHODS
4.4.1.1	SERVER-SIDE VALIDATION
4.4.1.2	CLIENT-SIDE VALIDATION
4.4.2	VALIDATION TYPES
4.4.3	VALIDATION FEEDBACK

4.5	DOCUMENT UPLOAD, STORAGE AND MANAGEMENT
4.5.1	DOCUMENT UPLOAD
4.5.1.1	TYPICAL USE-CASE SCENARIO
4.5.1.2	ISSUES AND CONCERNS TO BE ADDRESSED
4.5.1.3	FUNCTIONAL AND OPERATIONAL ISSUES IN FILE UPLOAD PROCESS
4.5.1.3.1	ADVISORIES REGARDING IMAGE FILE UPLOAD
4.5.1.3.2	RESTRICTIONS ON DOCUMENT TYPE AND EXTENSION
4.5.1.3.3	RESTRICTION ON FILE SIZE:
4.5.1.3.3.1	SIZE RESTRICTIONS FOR IMAGE FILE
4.5.1.3.3.2	FILE RESIZING
4.5.1.4	ISSUES TO BE ADDRESSED : SECURITY VULNERABILITIES
4.5.1.4.1	TYPICAL FORMS OF MALICIOUS ATTACK
4.5.1.4.2	CONSEQUENCES OF A MALICIOUS ATTACK:
4.5.1.4.3	DEFENDING AGAINST FILE UPLOAD ATTACKS
4.5.2	STORAGE OPTIONS : DATABASE v/s FILE SYSTEM
4.5.2.1	RELATIONAL DB
4.5.2.2	FILE SYSTEM
4.5.2.3	NOSQL DB
4.5.3	DOCUMENT MANAGEMENT

FORMS

The e-governance applications support large range of activities from providing information like location of RTO to providing intelligent GPS based accident reporting and support services. The effective implementation of e-Governance solutions requires hassle free interaction between e-governance application and its users. The “Form” is one of the most important element used in these systems/applications for interaction and collection of data. The effectiveness, efficiency and user satisfaction can be improved by developing applications with well-designed forms. To improve the usability of e-Governance Applications, it is important that sufficient attention is given to design and development of these web forms.

This section provides practical guidelines and best practices for design of web forms as part of e-governance applications to provide effectiveness, efficiency, and a pleasant and satisfying experience to the users who use them. The application of these guidelines will also help in reducing on design efforts and keep the consistency for end-user with more predictable and standardized user interface.

4.1

User-Centric Approach

As mentioned earlier, Web forms are one of the most important element in E-Governance applications. Despite their importance, the design of forms is often poorly thought out and conceived. While designing a form, we should look outside-in means the User should be kept in mind while designing the form. The primary goal while designing a form need to be “Speed” and “Accuracy”.

Web Forms must offer simple, intuitive and responsive user interfaces that let the users get things done with less effort and time. To achieve this, following should be kept in mind:

- Make the experience feel familiar. The users want experience to be familiar and comfortable. We should always try to use expected and standardised patterns.
- Understand Users/Stakeholders and their expectations from application. What is it that the stakeholder/user wants to do with our application? This user-centric approach is different from the one followed by most of us while designing applications “what we want users to do.” If our design makes it difficult for the user to feel successful, it make it difficult for our application to achieve its objectives.
- Practice appropriate consistency. We need to be consistent with our design. We can manipulate design to call attention to text or images, use design to provide

consistency, and to make disparate elements feel like a whole. For example, conventions as mentioned in GIGW guidelines should be followed in web applications also, wherever applicable. Besides, we should also establish conventions within our own project, such as the design grid, color pattern, typefaces used etc.

- Our goal should be to make the experience feel frictionless without the cognitive overload that leads to frustration and failure.

Despite differences in purpose, functionality and layout, all forms have three main aspects as noted by Caroline Jarrett and Gerry Gaffney in their book “Forms That Work: Designing Web Forms for Usability:

- Relationship: Forms establish a relationship between the user and the organization.
- Conversation: They establish a dialogue between the user and the organization.
- Appearance: By the way they look, they establish a relationship and a conversation.

For a form to be usable, all three aspects need to be considered. The following section briefly discuss about these three important aspects which are mostly ignored while designing forms for e-Governance applications.

4.1.1 RELATIONSHIP

A form is a means to establish or enhance a relationship between the user and the application owner. With this in mind, following principles shall be kept in mind while designing forms for applications:

- Every relationship has a goal, we need to be clear about the goal of our form before its design.
- Always base the title/name of the form on its purpose. This will clearly inform the user what the form is about and why they are filling it.
- Ask appropriate, timely questions and try to instil natural flow to form
- Consider scope of the form and do not ask data/questions beyond the scope of the project
- Choose appropriate language which user can understand
- Standardise the appearance and behaviour of form

4.1.2 CONVERSATION

A form is a conversation. And like a conversation, it represents two-way communication between user and application. The form should be considered as serving the interest of not only the application owner i.e. Government but also of the user.

The following guidelines help in achieving this objective:

- A form is a conversation, not an interrogation. Aggressive wording in labels should be avoided.
- Order the Form logically, reflecting the natural flow of a conversation. Details should be asked logically from user's perspective, and not the application or database logic. For example, asking the name after Date of Birth. It is better to ask more involved questions towards the end of the form.
- Group related information, such as personal details. The flow from one set of questions to the next should better resemble a conversation.
- As in a real conversation, each label should address one topic at a time, helping the user to respond in the corresponding input field.
- The natural pauses in a conversation will indicate where to introduce white space, how to group labels and whether to break the form up over multiple pages. Break the long forms by subject/Topic.
- In any conversation, people get distracted by background noise. So, remove clutter such as banners and unnecessary navigation that might distract users from filling out the form.
- Ensure consistent communication
- It is advisable to set user's expectation by using startup page and providing details on objectives for large forms requiring large information and time.
- For forms requiring external information like PAN Number, Aadhaar Card Number, Qualification details like %age of marks, photograph etc. provide the details before hand in a startup page and advise user to initiate with all information in hand.

4.1.3 APPEARANCE

The appearance or user interface (UI) is central to the usability of a Web form. Some of the general guidelines for appearance of the form are:

- Make form look good by harmonizing the placement of form elements
- The text on the form should be legible and use appropriate fonts
- The Field labels on forms should clearly explain what entries are desired
- There should be clear distinction between "required" and "optional" fields on form
- Text boxes on form should be of right length for expected response

- Use Cascading Style Sheet to provide standardised appearance to form

There are specific guidelines with respect to the elements like Labels, Input fields, Actions, Hints & Help etc. which are covered in detail in a separate section due to its important in form design.

4.2

Structuring of Form

As mentioned earlier, A form is a conversation. And like any conversation, it should be represented by a logical communication between two parties—User and the Owner of the Application. This section provides some important guidelines and best practices that can be followed for structuring of our forms. Some points are repeated to reinforce their importance.

ONLY ASK WHAT'S REQUIRED

We should only ask what we really need to meet the objective of our application. Every extra field we add to a form will affect its conversion rate. That's why we should always question why and how the information we request from users is being used.

ORDER THE FORM LOGICALLY

Details should be asked logically from a user's perspective, and not from the perspective of application or database logic. We should not ask date of birth before the name of the person.

GROUP RELATED INFORMATION

We should group related information in logical blocks or sets. The flow from one set of questions to the next should better resemble a conversation. Grouping related fields together also helps users make sense of the information that they must fill in.

Above is an example:

The better way of presenting this information is by grouping related information as depicted below:

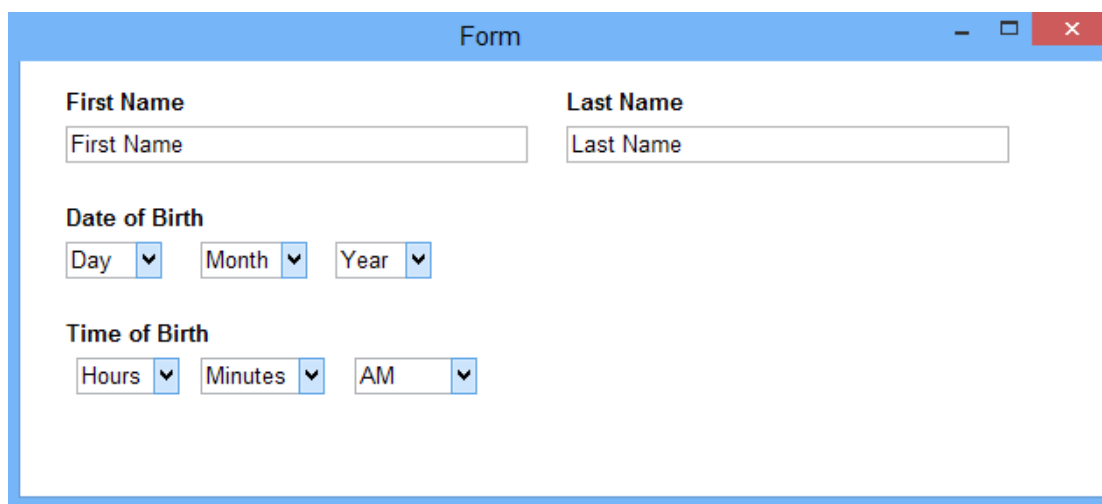
(Related information is grouped)

Use single Column instead of Multiple Columns

As per the best practices and usability guidelines, Forms should usually consist of one column. One of the problems with form fields in multiple columns is that the users may interpret the fields inconsistently. If a form has horizontally adjacent fields, the user will be required to scan in Z patterns, slowing the speed of comprehension.

MINIMIZE NUMBER OF FIELDS

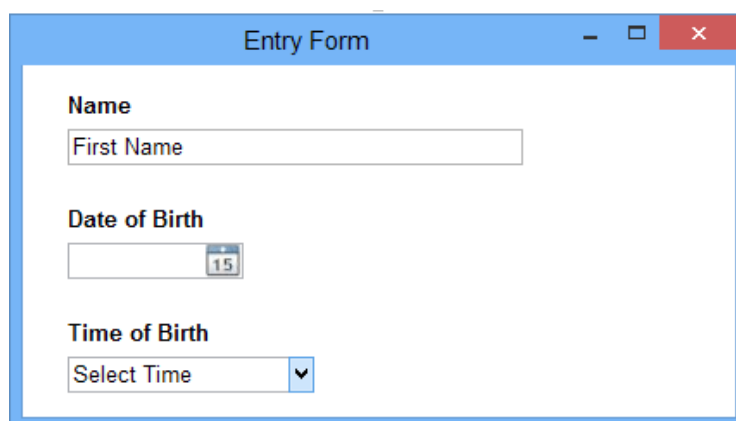
We should try to minimize the number of fields as much as possible to make the form less loaded. Wherever possible, try to combine multiple fields in one easy to fill field. For example, rather than asking Date of Birth as 'Day', 'Month' and 'Year' separately, it is better to ask Date of Birth as single field with Date Picker.



The screenshot shows a window titled "Form" with standard Windows window controls (minimize, maximize, close). Inside the window, there are three sections of form fields:

- First Name**: A text input field with the placeholder text "First Name".
- Last Name**: A text input field with the placeholder text "Last Name".
- Date of Birth**: Three separate dropdown menus labeled "Day", "Month", and "Year".
- Time of Birth**: Two dropdown menus labeled "Hours" and "Minutes", followed by a radio button group with "AM" and "PM" options.

Here the data is captured in multiple fields. Better option is to use Date Picker instead of using multiple fields.



The screenshot shows a window titled "Entry Form" with standard Windows window controls. Inside the window, there are three sections of form fields:

- Name**: A single text input field with the placeholder text "First Name".
- Date of Birth**: A single date picker field showing the date "15".
- Time of Birth**: A single dropdown menu with the placeholder text "Select Time".

CLEARLY MARK REQUIRED (MANDATORY) FIELDS

We should clearly distinguish between the mandatory and optional fields. The form should clearly mark the fields which cannot be left blank by the user. The convention is to use an asterisk (*) or 'Required'. For long forms with mostly required fields, we can use 'optional' to mark optional fields.

Register As*

Already Registered With ☒ None ☐ Employment Exchange ☐ Skill Providing

First Name*

Middle Name

Last Name

Gender* ☒ Male ☐ Female ☐ Transgender

Date of Birth*

Guardian/Father's Name*

(Mandatory fields are marked with asterisk)

SETTING DEFAULT VALUES

We should avoid using a default value in the field unless we are sure that large number of users (more than 90%) will be selecting that value. We should be very careful by providing default values in case of mandatory fields as user may simply skip with the default value and may not provide the desired value.

If the “Defaults” are used smartly in the form, they can make the completion of form faster with better accuracy. For example, based on geo-location of data, Country and City can be populated to make entry faster.

KEYBOARD-FRIENDLY FORMS

The users should be able to Tab through the form using keyboard also as number of users are more comfortable using key boards. This also makes navigation faster. We should take special care for large forms with lot of data entry as data entry operators rely more on keyboards navigation. We should also ensure that user are able to edit any field using key board only.

AUTOFOCUS FOR INPUT FIELD

Autofocusing at field gives users an indication and a starting point from where to start entry. We should provide a clear visual ‘notification’ on the focused field by using mechanism like changing color, fade in a box or highlighting the border of the field.

(The First Name Field is currently having auto focus in the diagram above. The field is clearly visible with different border and colour with Tab in the text box.)

Use Multi-Step Forms instead of Single Step Forms for large and complex forms

As per research on usability, dividing forms into multiple steps increase its usability and completion because:

- The first impression is less intimidating than a long form with lots of question fields.
- It has been observed that users are more likely to fill sensitive information like email, mobile number etc. on the final step
- The progress bar gives a satisfaction and motivation to complete the form.

(The Form above very clearly display by changing the colour (blue) in menu to provide detail about the stage)

Complex forms should be split over multiple pages with continue buttons and intelligent data saving. The users should be provided with progress bar to show where they are at in the process. The navigation between the pages should be smooth in both directions till the form is finally submitted. We should provide option to save the data for each subpage.

SHORTEN FORMS BY USING CONDITIONAL LOGIC

Conditional logic (sometimes called ‘branch logic’) is where we display a field based on response given in earlier field. For example, if the user select his type as ‘Student’, then only we should ask for his Degree, College Name and its Address.

This helps in reducing the length of our form and not displaying irrelevant fields to certain class of users.

TOP-LEFT ALIGNED LABELS ARE BEST FOR READABILITY & COMPLETION

Aligning labels just above fields on the left-hand side increases form completion time as per usability research as it requires fewer ‘visual fixations’. The Top-aligned labels are also better for forms with localization requirements as this allows labels with longer language to expand without disturbing the form structure.

PROGRESS INDICATOR

When the fields that need to be filled are spread across multiple Web pages, it is useful to communicate the status of progress through the form by clearly indicating scope, status and current position.

4.3

Form Elements

The Web Form is made up of elements. Every Web Form has at least three basic elements: labels, input fields, and actions. Labels is the mechanism used for asking questions. Input fields provide a way for users to answer those questions. Actions allow users to submit the answers they have provided. This section provides best practices and guidelines for Form elements.

4.3.1 LABELS

Labels are used to make the user interface more accessible by providing meaning to other form elements. The labels are provided to identify form controls including text fields, check boxes, radio buttons, drop-down etc. Labels need to describe the purpose of the form control.

This section provides guidelines for using Labels for better form usability.

LENGTH OF LABELS

Labels should not be used as replacement to Help Text. We should use descriptive and short labels (upto two words) to facilitate users to quickly scan through the form. The labels should clearly, concisely, and unambiguously define the required entry. Make labels distinct enough so that users do not confuse them with the data entries themselves. This can be done by bolding the labels or providing other visual cues, such as an asterisk.

WHETHER TO USE TITLE CASE OR SENTENCE CASE?

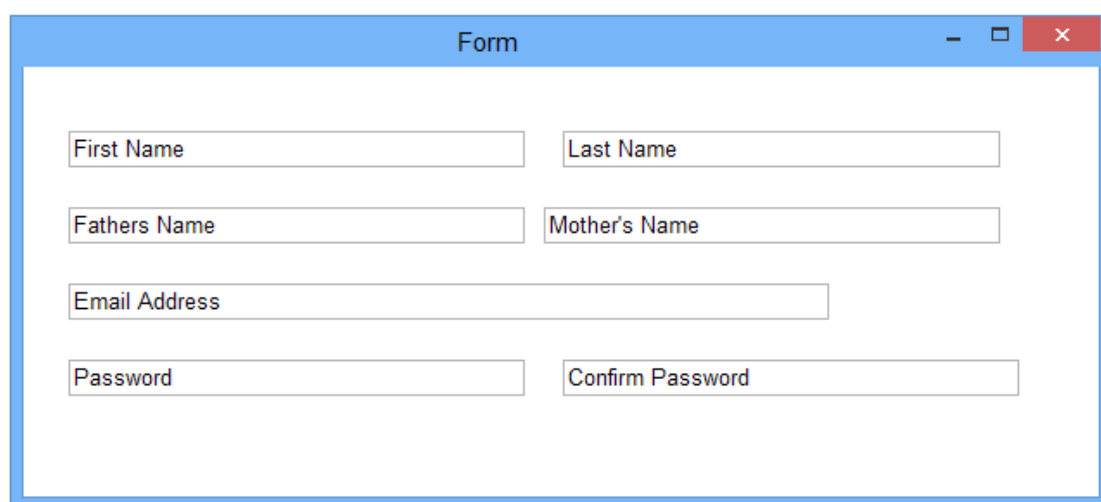
As per study, All Caps are difficult to read and we should avoid using all capital letters as Labels. Sentence case labels are slightly easier (and thus faster) for reading than title case.

ALIGNMENT OF LABELS

The aim should be to maintain a close and distinct visual relationship between the label and the form control. A study on form usability on label placement implies that forms are completed faster if the labels are on top of the fields. They facilitate faster scanning of forms. This also helps in handling flexible label length for localization.

INLINE LABELS (PLACEHOLDER TEXT)

For simple and small forms like Login Forms, inline labels can be used. For forms with more information, they are poor choice because the label disappear after user click on the field and user cannot double check. The inline labels are also confusing as the user may assume the field as already filled.



The image shows a web form window titled "Form". It contains the following fields:

- First Name
- Last Name
- Fathers Name
- Mother's Name
- Email Address
- Password
- Confirm Password

As the Inline Labels look good in form, we should use floating label along with placeholder label. The placeholder text is shown by default, but once an input field is tapped and text is entered the placeholder text fades out and a top aligned label animates in.

4.3.2 INPUT FIELDS

Once we know what data to be captured using the form, the input fields provide mechanism for the users to interact and provide the information. The different type of input fields used for this purpose are: Text boxes, radio buttons, check boxes, list boxes, drop down lists, combo box and buttons. It is important that we know how to use these elements while designing our forms for making interaction efficient and user friendly.

This section provides best practices for using these elements effectively while designing forms.

4.3.2.1 Text Boxes

The Text Boxes in Form are used to get text input (letters, numerals, or symbols) from the user or display text. The Text box is generally used for editable text although it can be used for displaying text also. Text boxes can display single line or multiple lines and can wrap text to the size of the control. The text boxes provide a single format style for text entered in the element.

4.3.2.2 Radio Buttons

Radio buttons are used when we want user to select one- and just one - option from a set of alternatives. As radio buttons are mutually exclusive, they should have a default value selected. The use of Radio buttons prevent users from entering erroneous data as they are only presented with applicable choices. The following are some best practices to be followed while using Radio Buttons;

Logical ordering for options- We should list the options in logical order. One suggested order can be from 'most likely selected' to 'least' or from simple to complex. Alphabetical ordering is usually not recommended for language dependent forms.

Comprehensive and clearly distinct options- the vague and misleading labels for options can create problems for users. It is important that the option are comprehensive and clear so that average user is able to understand. Appropriate help should be provided for better understanding, if required.

Always have default selection- Radio buttons should have one option pre-selected by default. Try to select the most likely option as the default option. If there is option for the user not to select any of the option, consider providing a Radio Button with "None" option by providing explicit choice.

Register As* Jobseeker

Already Registered With ☒ None ☐ Employment Exchange ☐ Skill Providing Institute

First Name*

Middle Name

Last Name

Gender* ☒ Male ☐ Female ☐ Transgender

Vertical laying of list – It is advisable to lay out lists vertically with one choice per line. Horizontal placement of list are difficult to scan. The horizontal arrangement also make it confusing to relate the button with label.

Registration Form

Name

First Name

Date of Birth

15

Time of Birth

Hours Minutes AM

Gender

☒ Male ☐ Female ☐ Transgender

Label as Clickable – Radio buttons are difficult to click due to their small size. To enlarge clickable area, we can also make the Label also clickable to provide better user experience.

Use Radio Button instead of Drop down list for limited values– In case we have upto 6 options, we should consider using radio buttons instead of drop down menus. The users are able to view all the available options without any clicking. In view of this, Radio buttons have lower cognitive load because they make all options visible so that users can easily compare them.

Avoid Nesting Radio Buttons– We should avoid using Radio Button within other Radio Buttons. It is better to keep all options at the same level to avoid any confusion.

4.3.2.3 Check Boxes

Check boxes are used when there is a list of options and the user may select any number of choices, including zero, one, or several. In other words, each check box is independent of all other check boxes in the list, and checking one box doesn't uncheck the others.

Number of the best practices mentioned above for Radio Buttons are applicable to Check Boxes also. Following are some of the additional best practices to be followed while deciding and using check boxes:

Standard Visual representation – We should always use standard visual representation like small square that has X or checkmark when selected to avoid confusion.

Use Active wording for Label – We should use positive and active wording for check box labels to clearly provide option to the user on the selected state.

Use check box for Yes/No options in place of radio buttons – If there are only two options, and only one is applicable with Yes/No kind of values, we can use Check Box instead of Radio Button which facilitates turning a single option on or off. If meaning of cleared check box is not clear, we should use radio buttons instead of check box.

4.3.2.4 Dropdown Lists & Combo boxes

The Drop Down Lists and Combo Boxes allow users to select exactly one choice from two or more mutually exclusive options. Users can choose one and only one option. With a standard drop-down list, users are limited to choices in the list, but with a combo box they can enter a choice that isn't in the list. The Dropdown lists are better choice than radio button for long lists as they use less space and improve selection.

Following are the best practices for using the Dropdown Lists and Combo Boxes:

Sort list items in a logical order – The items in the drop down list should be sorted in logical order by placing most common option first or using alphabetical order and grouping related options together.

Place options that represent 'All' or 'None' at the beginning of the List – If there are options applicable in the list which represent 'All' or 'None', then the same may be placed at the beginning of the List regardless of its sorting order.

Enclose meta-options in parentheses- The meta-options in the list should be represented in parentheses. For example, If we have a situation where none of the option is applicable, we can represent it as "(None)" which describes that the option is not being used.

Don't have blank list items—use meta-options instead - Users don't know how to interpret blank items, whereas the meaning of meta-options is explicit.

4.3.2.5 List Boxes

With a list box, users can select from a set of values presented in a list that is always visible. With a single-selection list box, users select one item from a list of mutually exclusive values. With a multiple-selection list box, users select zero or more items from a list of values. List boxes can act as a set of radio buttons (allowing people to select exactly one choice from a set of mutually exclusive options) or as a set of checkboxes (allowing people to select any number of choices from a list of options).

4.3.3 ACTIONS

As mentioned earlier, the form exists with some objective. After the user has completed filling the form with the desired input fields, some action needs to be taken on the form to complete the action. The actions can be saving the draft of the form, submitting the form data to the application owner or canceling what has been done. The prime objective of the form is met only on completion of actions which are taken on the form.

The actions on the form can be of two types:

- Primary Action
- Secondary Action

The Primary actions are the buttons that perform the essential functionalities required to meet the main objective of the form. The Actions like Save, Submit or Continue enable in achieving the primary objective of the Form, they are referred as "Primary Action".

In contrast, Secondary actions are used to retract the work which has been carried out on the form. The actions like 'Cancel', 'Reset' or 'Back' are secondary actions which signals stopping the data entry and retracting back to the previous stage. The secondary actions are counter to the basic objective of the form and have negative consequences; they should be incorporated with care.

Some of the important points to consider while incorporating the Actions in the form are:

- Keep the primary actions as prominent in the form
- Avoid incorporating secondary actions wherever possible. If secondary actions are to be included, give them less visual weight than primary actions.
- To ensure that secondary actions do not result in loss of data, build default “Un Do” option as a choice.
- Display warning to the user regarding loss of data in case the secondary actions are performed on the form to give a chance to go back
- Consider placement of these Action buttons on the form. The Primary Action buttons should be placed aligned with Input fields and make it difficult for the user to use secondary action unknowingly.
- Replace the primary action button with message or animation to represent the submission in progress to avoid execution of primary action twice resulting in duplication of submission. For example, while file upload in progress, present an animation to inform the user about current status of the submission.
- It is also a good strategy to keep the Primary Action button disabled, till all the data is entered.
- For any legal requirement, combine terms and conditions with primary action.

4.3.4 HELP TEXT

As mentioned, the e-Governance applications are used by a wide range of users and are complex in nature. It is important that proper and timely help is provided to the user to complete the form to meet its objective. “Help Text” are a set of messages that provide instructions to help users understand how to complete the form and use individual elements. This includes expected value in the field, data format, mandatory or optional input, and any other help. “Help Text” are often associated with fields that are unfamiliar and help users to enter data properly. We should consider the following when providing help text in our web forms:

- Help text should be used only for unfamiliar input fields. As mentioned earlier, use familiar words and try to minimize the help required at a field.
- Concise help text, visible and adjacent to the question being asked, provides the most clarity for people.

The following are different ways in which help text can be incorporated in web forms:

4.3.4.1 Form level Instructions

Wherever relevant, provide overall instructions that apply to the entire form. This can include :

- Objective of form,

- Information to be kept handy before starting the entry like Aadhaar number, PAN Number,
- Data format like upload photograph as JPG for specific size,
- Time out limit, if any

It should be ensured that user read these instructions before starting entry of form. The instructions should be concise and clear.

Unique Identification Authority of India
Government of India

Aadhaar Self Service Update Portal

[Aadhaar Home](#) | [Instructions](#) | [Update Status](#) | [FAQ](#) | [Contact Us](#)

Dear Resident,

Welcome to Aadhaar Self Service Update Portal. UIDAI is committed to provide resident friendly eco-system and self service update portal is one of the various touch points to update their profile in 3 easy steps: **STEP 1: Login with Aadhaar**, **STEP 2: Upload Documents** and **STEP 3: Select BPO Service Provide and submit request**. Note that Update here refers to any changes as well as corrections, if required, in resident's original Aadhaar letter.

Who can use this portal?

Any resident with a mobile number can update their profile using this portal. **Mobile number is mandatory to receive password for login.**

What all information can be updated through this portal?

Residents can update Name, Address, Gender, Date of Birth, Mobile Number and Email ID through this portal. For other updates, please visit Aadhaar Enrollment / Update Center. [Click here](#) for detailed instructions on submitting Aadhaar Update/ Correction requests through this portal.

No documents required to be submitted for Mobile and Gender correction.

What are the documents required to be submitted along with an update request?

Depending on the field to be updated, attach original scanned (with color scanner) copies of supporting documents as per the Valid Documents List. Please [Refer Link](#) for valid documents.

Will resident information be Updated immediately after submission of request?

Submission of information for update does not guarantee update of Aadhaar data. The information submitted is subject to verification and validation.

To submit your update/ correction request online please [CLICK HERE](#)
To submit your update/ correction request by post please [CLICK HERE](#) to download the correction form

Copyright © 2012 UIDAI All Rights Reserved. This website is best viewed in 1024x768 screen resolution.

4.3.4.2 Inline Instructions

Inline instructions are provided to assist the user while using the form. These instructions can be incorporated in the following ways:

- Provide instructions with in Label. For example Date (DD/MM/YYYY), Name (Required)

- Provide outside label by placing an icon next to the field which user can click if help is required. A “Question Mark” is most common icon used as visual cue for user to trigger this help
- Even better, provide help dynamically using “Dynamic Popups”. The “Dynamic Pops” appear when the user clicks or hover on the field
- Using Placeholder text instructions can be provided inside the form field. This can be used for providing example of expected data and expected format. Display them with lower colour contrast than text entered by user for clear demarcation. As discussed earlier, placeholder text is not replacement for Labels. This should also be kept in mind that once data is entered, the instructions are not available for verification and review before final submission of data.

Each of these options has its distinct advantages and disadvantages. As with all design decisions, an understanding of user needs and business goals should inform which of the option is best suited for our requirement.

4.4

Form Validations

A well designed web form is combination of user interface and data validation. The earlier section has dealt the subject of User Interface. The form objective is only met when it ensures successful submission of error free data. The proper validations ensures that the user provide necessary and properly formatted information needed to successfully complete an operation.

The Validations also play a very important role of not only securing the data but also protect against malicious activity. They also guide user and ensure painless and quick entry of data. By looking at its importance, it is important that we incorporate them properly in our web forms.

The objective of most of e-Governance applications is to capture data from users for providing various services. It is important that the quality of data is ensured. As mostly the data captured is further used for policy planning and decision making, the poor quality of data can results in ill-informed decisions and policies. Validations are very important tool to ensure quality of data captured as part of our applications.

This section explores various methods and type of validations and provides best practices and guidelines for building proper validations in the form.

4.4.1 VALIDATION METHODS

The input provided by the user in the web form can be validated on the server and on the client using server side validation and client-side validation.

4.4.1.1 Server-side validation

After the form is completed by the user and submitted, the data/information is sent to the server for saving. The information is then validated at the server using server side validation language/script before its saving using the server-side validation. If validation fails, the response with errors is sent back to the client for information and taking necessary corrective measures. The server side validation is important as it ensures that validations are not bypassed by the user. The drawback is that user have to complete the data entry without getting any response. This can become frustrating for the user especially in large forms if action is not successful.

4.4.1.2 Client-side validation

Server-side validation is sufficient to have a successful and secure form validation. However for better user experience, we need to consider using client-side validation. In client-side validation, the validation is done on the client i.e. Web Browser using script language such as JavaScript. By using JavaScript, user's input can be validated as it is entered. This provides a more responsive experience to the user resulting in user satisfaction. With client-side validation, form never gets submitted if validation fails.

Main drawback of client-side validation is its reliance on client side scripting language which can be controlled by the user. If JavaScript is turned off in the browser either deliberately or un-intentionally, the validation can be bypassed easily.

This is why we should combine and implement validations on both client and server. The combination of both the methods provide faster response, more secure validation and better user experience.

4.4.2 VALIDATION TYPES

The following are different type of validations which are performed to ensure error free information submission.

REQUIRED FIELD

This validation ensures that user does not skip any mandatory field in the form. The data field is validated to ensure that action does not complete without these mandatory parameters. Mandatory/Required fields should be clearly marked in order to inform users about what information has to be provided up front as discussed earlier. It is also a good practice to put a note on top of the form to indicate that fields marked with asterisk are required fields.

CORRECT FORMAT

The validation also has to ensure that users provide information in correct format. These validations are performed by checking the entry made against a defined pattern. This enables

to check for predictable sequence of characters like email address, telephone numbers, PAN Number which has defined pattern. We should try not impose strict pattern on the users as this becomes hindrance to speedy completion of entry. Wherever possible, the system should intelligently convert to the right format, if data is not entered in expected format.

CHECKING FOR RANGE OF VALUES

These validations are performed to check that the information entered is between specified lower and upper boundaries. We can check ranges within pairs of numbers, alphabetic characters, and dates.

CONFIRMATION FIELDS

For important data, it is a good practice to let the users confirm their input using additional confirmation fields. This ensures that user provides correct information. The probable cases are Password, entry of Credit card number, Aadhaar Number entry etc. A confirmation field should be placed next (or below) the target field. It has to clearly describe the purpose of the field such as “Confirm your password”. If two values do not match, the user should be informed.

BUSINESS RULES VALIDATIONS TO ENSURE DATA QUALITY

As discussed earlier, poor quality of data has long term implications. To ensure, quality of data captured as part of our application, the validations related to business rules should be planned carefully. For example, if Date of Joining and Date of Birth is captured, the date of joining cannot be greater than Date of Birth. Also Date of Joining should be within the range of 60 years. The business rule to be applicable on each field should be considered properly as part of the application design. The relationship of elements values within the form/record should be considered and enforced. Accuracy and Validity of data is to be ensured before it is saved in the database.

The business rule should be checked and validated both at client end and at server end as per its applicability.

Refer to “Data Quality” section for details on various rules to be enforced on standard data elements captured as part of e-Governance applications for data quality.

VALIDATIONS FOR SECURE WEB FORMS

As mentioned, web forms are interface of user with application. The web forms without proper validations can be exploited by unauthorised or malicious users. To make our web form secure following should be ensured:

- Put reasonable limitation on number of characters which can be entered in Input Field. For example, to enter name we can allow user to enter approximately 50 characters.

Without proper input restriction, a hacker may try to send large amount of data with a intention to crash the system.

- Limit the possible characters which can be entered in the field. For example, name should not allow any special character. The special characters are used by intruders to send and execute scripts through these input fields.
- Secure form from cross side scripting (XSS) : Cross-site scripting (XSS) is a code injection attack that allows an attacker to execute malicious JavaScript in another user's browser. The malicious user exploits the vulnerabilities in the web form to execute scripts at the client. The attacker can use XSS to perform the following attacks:
 - o Keylogging : The attack can register a key board event listerner and can then send the key strokes from users computer to his server. This can result in exposing sensitive information like Passwords.
 - o Cookie Theft: The user attacker can get access to the cookies associated with the application and use them to extract sensitive information like session ID.
 - o Phishing: The attacker can insert a fake login form and send the submitted information like user ID/password through this form to his server.

To secure application from XSS attack, the following mechanism should be used for securing the input:

- Encoding, which escapes the user input so that the browser interprets it only as data, not as code.
- Validation, which filters the user input so that the browser interprets it as code without malicious commands.

As mentioned earlier, the validations should be performed at both client and server to escape XSS attack.

- Secure against Sql Injection: SQL Injection is an injection attack where an attacker can execute malicious SQL statements on database server of web application. By leveraging an SQL Injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database. SQL Injection can also be used to add, modify and delete records in a database, affecting data integrity.

Proper validation of data needs to be ensured at both client and server to handle these vulnerabilities. Please refer to various guidelines available on <http://security.nic.in/> under developer's section.

4.4.3 VALIDATION FEEDBACK

If validation fails, the system should let the users know it by providing a clear and unambiguous message (usually one or two sentences) and how to correct errors. It is good

practice to place error message at the top of the form, before all the other fields for better visibility. This also allows screen readers to easily access these messages.

As a practice, the message should preferably be in red colour with appropriate icon like red circle with white space. The users should be clearly informed about the fields which needs attention with concise message.

The fields with errors should be clearly marked using

- Providing red inline messages or markers next to every invalid field
- changing background color or border color of invalid fields (to red)
- changing the color of field labels
- providing error tips (balloons) next to each field

Help to correct the error should be short and clear. For example, if date is in incorrect format, mention error message as “The date should be in the dd/mm/yyyy-format”. The error messages can be provided using :

VALIDATION ON SUBMISSION

In this, the form validation is performed when the user submits the data after completing the data entry. The errors are provided on all the fields in one go. This allows user to fill information without any interruption but user are able to fix errors only when they try to submit the data.

As a practice, this validation is usually used for server side validations including where the validation is to be performed against existing database like valid State Code, valid user ID etc. This can also be used for final validation at client end before submission of data. As practice, avoid displaying errors on a separate page. The users will be require to shuffle between web form and error page to fix all the errors.

REAL-TIME VALIDATION (OR INSTANT VALIDATION)

Real-time-validation are performed in real time while entering the data. The objective is to immediately alert the user while the data entry is made. instant validation occurs during typing in an input field or after the input field loses focus.

Instant validation should be implemented carefully and in appropriate cases because it might be distracting if overused or misused. The real-time validation is performed at the client end using client side scripting. AJAX can be used to perform real-time-validation from server. For example, user ID can be validated from server on entry of the value in the field.

MINIMIZING ERRORS

Besides validation, the objective should be to minimize the errors by facilitating the users.

The “Help Text” discussed in the previous section can be used effectively to minimize errors by providing proper instructions to the user.

Apart from validating user’s input and providing Help to the users, the application can force users to enter data in a particular format and also convert data to the desired format programmatically. This can be done by masking input fields in order to force users to enter information in an appropriate format. Mask is an expression that controls what users can enter in an input field.

The above section presented different methods and techniques that can be used to implement validation in forms. Although there are many possibilities, one should carefully plan validation for each project. Not all techniques provide a solution for everything. The following are some basic guidelines for web form validation design:

- Never omit server-side validation.
- Don’t provide confusing validation feedback. It should clearly communicate the errors and ways to fix them.
- Don’t let users think about what information is required, always clearly mark required fields.
- Don’t provide all validation feedback on a single page or as a popup alert.
- Don’t use dynamic effects as compensation for a badly designed form. Fancy effects won’t hide a poorly designed web form.
- Don’t forget to inform users when the form was completed successfully. It is as important as a good validation feedback.

4.5

Document Upload, Storage and Management

The requirement to upload files on a portal/ web-application is a common feature, mostly used to submit documents and images on an Online System. With increasing thrust on paperless file movement, document upload and retrieval as part of the workflow has become ubiquitous. Also, many citizen-centric applications require online submission of documents as part of the application submission process.

Documents, images uploaded on a web application become integral part of the application work-flow with multiple users with different roles accessing them at different stages of the document life-cycle (creation, uploading, storage, tracking, retrieval, disposal etc). Keeping in view the limitation of bandwidth and need for faster upload/ retrieval speed, document size, formats etc need to be carefully planned and controlled. Since documents are fairly

large in size compared to structured data, storage planning and management is another critical part. Very quickly, any large allocation of storage infra-structure can run out of space, if proper planning and implementation with respect to file type & size, relocation & disposal policy etc are not put in place. Further, from the cyber-security stand-point, and the ever-hanging threat of malicious attacks, all possible safeguard rules and procedures should be built into the document management process.

4.5.1 DOCUMENT UPLOAD

4.5.1.1 Typical Use-Case Scenario

A web application may typically be designed with the option for uploading a number of documents as part of the business requirement. For example, a citizen-centric application form – say, request for Driving License - may require the following categories of documents:

- Proof of Identity
- Proof of Address/Residence
- Proof of Age
- Proof of Educational Qualification
- Photograph
- Signature

For each category, there may be a permissible list of documents out of which, the applicant may choose one as per availability/ convenience. For example, for Identity proof, the options may include:

- Aadhaar Number/card
- Passport
- ECI Voter card,
- Office ID, etc.

The web application may present the category-wise list of options in a combo box. Applicant will select one of these possible options and then upload the scanned copy of the related document.

In a service centre context (like Passport Seva, Parivahan Seva etc), the application may get an added integration with a document capturing system (like scanner, MFD) which may be made part of the work-flow and make available the scanned copy of the documents to the subsequent stage of pre-verification or pre-processing before upload.

4.5.1.2 Issues and Concerns to be Addressed

There are two broad areas of concern that need to be addressed when we design a web application or portal involving file upload needs:

- Functional and operational aspect: It pertains to the need of the system to be lean & agile, facilitate faster upload and retrieval, optimize bandwidth & storage space. This affects the over-all operational efficiency and cost.
- Security aspect: It concerns allowable file types, content checking, restrictions on extensions, storage safeguards and so on.

4.5.1.3 Functional and Operational issues in File Upload Process

These issues are handled through prior planning the specific requirements/ business rules etc., creation of a set of policies, rules, restrictions and advisories, and implementation them in the application/ Document Manage System. Some of the rules/ restrictions pertain to File Formats and Extensions, Naming conventions, File size, Optimization options etc.

4.5.1.3.1 Advisories Regarding Image File Upload

Image file upload in itself, is a considerably broad functional area and needs elaborate discussion:

- Almost any image, apart from photographs and scanned images, should preferably use Scalable Vector Graphics (SVG) format. The biggest advantage of such images is that they scale easily and without any loss of visible distortions and hence image size is not dependant on bigger byte size. Drawings, icons, logos, maps, flags and other such images are preferably uploaded in SVG format as vector images. Almost all major browsers (including Chrome, Firefox, IE, Edge, opera, Safari etc) have SVG rendering support.
- JPEG is preferable for images comprising thousands of colour shades/ tones where the color changes fluidly throughout the image, as in a photograph. Photos and scanned images should be converted into JPEG format for better space optimization. If you can find an original of a photograph in 16-bit or 24-bit PNG or TIFF, edit that, and save as JPEG before you upload. It may be noted that JPEG is a lossy image format and multiple rounds of editing may degrade image quality.
- PNG files are good for images with relatively few colors, such as a drawing, sketch, flag, chart, map etc. Its compression is optimized for images with large areas of identical color with sharp edges. PNG uses lossless compression that allow further editing without degrading the image.
- Inline animations are best stored in animated GIF format. For Video, Ogg, Theora or WebM formats are preferable.
- Depending on image characteristics, changing format from one type to another (say .jpeg to .png to .svg or the other way round) may provide better quality for a lesser storage space. If you have a good image that is in the wrong format, convert it to

the correct format before uploading. However, if you find a map, flag, etc. in JPEG format, converting it to PNG is likely to reduce file size.

4.5.1.3.2 Restrictions on Document Type and Extension

For a particular category of document (say for proof of address, or photograph), only specified file formats and extensions may be allowed. The permissible extensions need to be white-listed and anything outside of this list will be rejected. Depending on specific business requirements, the permissible file types/ extensions may vary:

- Documents (for purpose of proving address, identity, age etc):.pdf, .doc, .docx, .txt etc.
- Data : .xls, .xlsx etc.
- Image (photograph, maps, pictures etc.) : .jpeg/.jpg, .png, .svg etc.

4.5.1.3.3 Restriction on File Size:

Size restriction is can be imposed on the scanned document, images before upload. Since the document uploaded is normally integrated with the workflow, it may be required to be downloaded/ viewed by a user at a subsequent stage. File size should be sufficient for the specific purpose, without compromising quality. An unnecessarily large file will create load on the network. Further, the scanned documents/ images need huge storage space over a period of time and hence need optimization. Appropriate restrictions and specifications on the uploaded size of documents/ image files will enable efficient resource management and enhancement of speed/ performance. Depending on business need, the following restrictions of maximum file size may be imposed:

- For PDF documents: 500 KB per page (can be reduced to 200 KB too)
- For ID photographs: 100 KB (can be reduced down to 20 KB too)

4.5.1.3.3.1 Size restrictions for image file

Image file size restriction is generally imposed on two parameters:

- Visual size (size in terms of pixels)
- Storage size (size in terms of KB/MB)

4.5.1.3.3.1.1 Size in terms of pixels

Typically, a photograph may be specified in terms of width x height. For example, maximum size of a passport-size photo may be 110 pixel x 140 pixel. A signature scan may be specified with a maximum size of say, 140 pixel x 110 pixel.

4.5.1.3.3.1.2 Size in terms of storage space

Typically, maximum size of a passport size photograph may not be allowed to exceed 100 KB (even 20-30 KB passport-size photos are also acceptable).

4.5.1.3.3.2 File Resizing

A file may need to be down-sized from the original form before upload in order to fit into the business/legal/technical requirement of the web application. For a web application where file/image upload functionality is given, and specific file size, format etc are specified, it is helpful to provide a built-in image resizing tool, or at least a link to an online resizing service, so that users may not require any additional software for meeting the requirements. Resizing tool facilitates image editing and re-formatting options like cropping, colour reduction, compression, image rotation, adding text and do on. Cropping helps in removing irrelevant areas from an image. A number of open source resizing and image editing tools are available which can be utilized to ensure better user facilitation.

4.5.1.4 Issues to be Addressed : Security Vulnerabilities

The process of document upload and storage for any IT system has immense security implications since many malicious attacks take advantage of loopholes in this process and get entry into the system.

4.5.1.4.1 Typical forms of malicious attack

There are a number of ways a website can be attacked by a file upload, such as:

- A file could overwrite another existing file in the server having exactly the same name. This may cause
 - o The website to stop working
 - o The website to function incorrectly
 - o Get the website defaced
 - o To modify the list of allowed file types in order to make further attacks simpler.
- A malicious file containing damaging code could be uploaded and get executed through some further means. The uploaded file could contain malicious code in the form of an exploit, virus, Trojan or malware, which could be used to gain control of the Web server. For example, it is possible to hide PHP code inside an image file and still have it appear to be an image. When the image is opened, it also executes the code hidden in the file.
- The web server can be compromised by uploading and executing a web-shell which can run commands, browse system files, browse local resources, attack other servers, or exploit the local vulnerabilities, and so forth. These comprise server-side attacks

- The file could contain scripts or tags that exploit other well-known Web application vulnerabilities, such as cross-site scripting (XSS) and other client-side attacks.
- Uploaded files can be abused to exploit other vulnerable sections of an application when a file on the same or a trusted server is needed. This can lead to further client-side or server-side attacks.
- The file space of the Web server could be exhausted by the attacker uploading a huge file or multiple large files causing the application to drastically slow down or stop responding altogether. This will result in Denial of Service (DoS).
- If the uploaded file can be accessed by entering a specific URL path, it could be especially dangerous because the file could be executed immediately after uploading.

4.5.1.4.2 Consequences of a malicious attack:

Following are the possible consequences of malicious attacks that may occur through document upload process:

- Complete system takeover
- An overloaded file system or database causing denial of service to real users
- Forwarding attacks to back-end systems
- An attacker might put a phishing page into the website
- Simple defacement causing embarrassment
- Stealing of sensitive data
- Launch further attacks on other systems

4.5.1.4.3 Defending against file upload attacks

There are number of precautionary measures that need to be implemented in any web application/ website to protect itself from file-upload attacks. While these techniques cannot guarantee a website will never be attacked from a malicious file upload, they will go a long way toward protecting the website while still providing users with the benefits of uploading files when needed.

- The application should use a white-list of allowed file types. This list determines the types of files that can be uploaded, and rejects all files that do not match approved types. Only those file types and extensions that are necessary for business functionality should be allowed. Any script/program file such as .php, .jsp, .asp etc. should not be allowed. Uploading compressed file or archive should be disallowed unless essential. If allowed, contents of the compressed file should be checked individually.
- Any special character and more than one dot should not be allowed in the file name. File name and extension cannot be blank. The application should also set a maximum length for the file name.

- The application should use input validation to ensure evasion techniques have not been used to bypass the white-list filter. These evasion techniques could include appending a second file type to the file name (e.g. evilimage.php.jpg) or using trailing space or dots in the file name. Thorough filtering and content checking routine can identify a malicious file even if the file name and extensions are of approved type. For example, the MIME-type should be consistent with the file extension. Also an image file can be rendered to check its authenticity before it is stored in the system.
- It is recommended that the application should change the supplied file name to a different one as per pre-defined protocol or naming convention. Use of an algorithm to rename the files is also commonly followed. For instance, a filename can be a MD5 hash of the name of file plus the date. Prevent from overwriting a file in case of having the same hash for both.
- Whenever possible, uploaded files should be scanned by antivirus software. Either use a virus scanner on the server, or (if the contents of files are not confidential), a free virus scanner website can be used. In this case, file should be stored with a random name and without any extension on the server first, and after the virus checking (uploading to a free virus scanner website and getting back the result), it can be renamed to its specific name and extension. Further, the upload directory should be subjected to virus/malware checks at frequent intervals.
- The directory to which files are uploaded should be outside of the website root. Further, the directory should not have any “execute” permission and all the script handlers should be removed from these directories. Write permission should be removed from files and folders other than the upload folders.
- Limit the file size to a maximum value in order to prevent denial of service attacks. It is recommended to provide an Image Re-sizer tool in the application, particularly for upload of images.
- Use Cross Site Request Forgery protection methods. Adding the “Content-Disposition: Attachment” and “X-Content-Type-Options: nosniff” headers to the response of static files will secure the website against Flash or PDF-based cross-site content-hijacking attacks. Browser caching should be disabled for the corssdomain.xml and clientaccesspolicy.xml files.
- If very sensitive, and if file size/number is not very high, consider saving the files in a database rather than on the file system. However, this has got other technical implications.
- Ensure that configuration files such as “.htaccess” or “web.config” cannot be replaced using file uploaders. Ensure that appropriate settings are available to ignore the “.htaccess” or “web.config” files if uploaded in the upload directories.
- Ensure that uploaded files cannot be accessed by unauthorized users.

4.5.2 STORAGE OPTIONS : DATABASE V/S FILE SYSTEM

Once uploaded into the application, the possible options for storing the files/ images are as below:

- Storing in the Relational Database (RDB) as binary data
- Storing in the File system (FS)
- Storing in NoSQL DB options

4.5.2.1 Relational DB

Almost all RDBMS provide options to store documents or images as part of the database. Normally such data is stored as binary objects (BLOB in Oracle, IMAGE in SQLServer, BLOB in PostgreSQL etc). Although storage of images, documents etc. in the DB may, in some cases provide better data security and transactional integrity, however, this practice, in general, is not favoured in view of the huge load leading the RDBMS to lose its efficiency. Back-up and retrieval management also becomes a big challenge. Further, normal RDBMS is not optimized to handle unstructured data.

4.5.2.2 File System

Storage in File System is the normally advisable option unless specific requirement related to security and integrity of the document/ image is a major requirement.

For the File System option, an appropriate storage framework needs to be designed and created with a tree-structure of folders/ directories based on key metadata parameters do as to ensure proper placement and retrieval of specific document. A Document Management System (DMS) – which automates this process and also facilitates easy management, back-up, relocation etc. is recommended in case of a File System based storage option.

4.5.2.3 NoSQL DB

NoSQL Databases also provide a good option to manage unstructured documents and also the added benefit of horizontal scaling. NoSQL DB options include MongoDB, CouchBase, Redis, Elasticsearch and so on.

MongoDB is a document oriented database which provides high performance, availability and scalability. It provides a JSON like document structure with dynamic schema called BSON format. It also supports sharding, dynamic queries on documents using a document based query language. BSON objects can store any uploaded document like image file or a PDF document along with the metadata. It has a maximum capacity up to 16 MB which is sufficient for most purpose of document upload. Any object beyond this limit can also be stored in MongoDB by using GridFS API.

Sr. No.	Action	Conform
1.	Field labels on forms clearly explain what entries are desired	
2.	Labels are placed close to the data entry fields	
3.	Text boxes on forms are the right length for the expected response	
4.	When field labels on forms take the form of questions, the questions are stated in clear, simple language	
5.	Fields in data entry screens contain default values when appropriate and show the structure of the data and the field length	
6.	The “required” and “optional” fields on forms are clearly marked	
7.	The application automatically formats the entered data in the desired format	
8.	Forms pre-warn the user if external information is needed for completion (e.g. PAN Number, Aadhaar Number, passport number etc.)	
9.	Information on forms are grouped logically, and each group has a heading	
10.	Fields on forms contain hints, examples or model answers to demonstrate the expected input	
11.	Pull-down menus, radio buttons and check boxes are used in preference to text entry fields on forms (i.e. text entry fields are not overused)	
12.	The cursor is placed where the input is needed	
13.	Data formats are clearly indicated for input (e.g. dates)	
14.	Forms allow users to stay with a single interaction method for as long as possible (i.e. users do not need to make numerous shifts from keyboard to mouse to keyboard).	
15.	The application is carrying out the field level validation and form level validations at appropriate times	
16.	Both Server Side validation and Client side validations are incorporated appropriately	
17.	The care is taken to make the correction of errors easy	
18.	The business validation rules are reviewed and considered.	
19.	The intra-record dependency is considered through as part of the application	

4.5.3 DOCUMENT MANAGEMENT

Sr. No.	Top 10 List of do's and don'ts
1.	Decide on a file upload policy and Standard Operating Procedure (SOP) based on business requirements.
2.	Identify the different categories of documents required to be uploaded as per business need: <ul style="list-style-type: none"> • Proof of Identity • Proof of Address • Age Proof Document • Educational Qualification Certificate • Photograph • Signature • Etc.
3.	For each identified category, list the set of permissible documents (For example, for identity, the concerned user department may allow the following): <ul style="list-style-type: none"> • Aadhaar Number/card • Passport • ECI Voter card, • Office ID • Etc.
4.	Identify what file type (and extension) are to be allowed and white-list them type-wise. Sample permissible types are as below: <ul style="list-style-type: none"> • Documents → pdf, .doc, .docx, .txt etc. • Data → xls, .xlsx etc. • Image → .jpeg/.jpg, .png, .svg etc.
5.	Set limits for maximum permissible size for different file types. For image file like photograph, signature etc., additionally set allowed dimension in terms of pixel.
6.	Determine storage mechanism from the following options: <ul style="list-style-type: none"> • Store in File System (optionally with DMS support) • Store as binary object in RDBMS • Store in a NoSQL database
7.	Implement storage policy. If using file system, determine and configure storage media, drives, folder structures, permissions, file naming conventions, and so on. Also, determine back-up, relocation and weeding-out policy.

Sr. No.	Top 10 List of do's and don'ts
8.	<p>Provide automatic checking of uploaded files for</p> <ul style="list-style-type: none">• File extension• MIME Type• Whether file name contains special characters• Whether file name length is below maximum limit• Whether file name contains multiple extensions• If possible scan for virus/ malware• Check if file size is within permissible limit
9.	<p>Before storage of files, carry out necessary re-naming, re-sizing, compression, encoding etc. as per policy.</p>
10.	<p>Ensure that the uploaded file is stored along with meta-data comprising frequently used search parameters for easy reference and retrieval. Necessary index on the metadata parameters need to be implemented.</p>

- 5.1 INTRODUCTION
- 5.2 CONDUCT A THOROUGH ANALYSIS OF USER SCENARIOS
- 5.3 KNOW YOUR USER
- 5.4 SETTING UP A REPORT – THE QUERY FILTER
- 5.5 REPORT LAYOUT
- 5.6 EMPHASIZE IMPORTANT INFORMATION
- 5.7 FORMAT AND PAGINATE
- 5.8 MAKE THE REPORT DISTRIBUTABLE
- 5.9 DESIGN DATABASE SPECIFICALLY FOR REPORTS
 - 5.9.1 TRANSACTIONAL VS REPORTING DATABASE
 - 5.9.2 WHEN TO BUILD A SEPARATE REPORTING DATABASE?
 - 5.9.3 ADVANTAGES OF HAVING A SEPARATE REPORTING DATABASE
 - 5.9.4 BEST PRACTICES FOR BUILDING A REPORTING DATABASE
- 5.10 REPORTING FRAMEWORKS
- 5.11 CONCLUSION

REPORTS

5.1

Introduction

When we begin to build an application, the focus is naturally on designing input screens for capturing data and designing the database which is optimized for data entry. However, the main purpose of any application is not just to capture data but to use it to monitor performance, use it as a guide in decision making and developing future policy guidelines. The management primarily concentrates on what the stored data can reveal about a programme or performance of an entity and how they can use that information to improve the performance so that the overall governance objective is effectively met. Reporting and analytics are two areas which fulfil these aspirations of the decision makers.

This document gives you a few guidelines and checklist for preparation of reports for your application. The first step is obviously understanding the user requirements. The first section identifies a set of questions you should ask your client or the end user and also yourself before you build report. Rest of the sections elaborate on the questions by giving you simple guidelines for generating useful and user-friendly reports.

5.2

Conduct a thorough analysis of user scenarios

Before building a report, it is important to understand thoroughly, the client's reporting requirements. Some of the questions which should be asked while capturing the user requirements include:

- a. Who will be using the report?
- b. What parameters will be required to customize the report as per user's requirements?
- c. What all information is required as output of the report?
- d. Are there any important information in the report which will need to be highlighted?
- e. Is there any data which should not be shown in the report?
- f. If the report is designed to display calculated values, then what is the exact formula for calculating the same?

- g. If drill down functionality is to be provided, then which parameter should be used to drill down and what should be the output of the report on drilling down?
- h. How will the user want the report to be viewed?

5.3

Know your user

The most important question to answer before building a report is who is going to be the user of the report. Different users have different information needs. There are usually four types of users who use the reports generated from an e-Governance solution. They are:

- Operational Users – Users who are responsible for entering the operational data in the system.
- Middle-level Management - The district collectors invariably form the middle-level management.
- Senior Management - This includes the officials at the level of Joint Secretaries in GoI and Principal Secretaries in the State who are instrumental in formulating policies and laying down implementation guidelines.
- Public – This may include inter alia general public, research students, NGOs etc

Each of the above users have their own reporting requirements:

- **Operational Users**
 - Interested in viewing data entered by them
 - May view the data from different perspectives
 - Need simple tabular reports
 - Not interested in long term analysis or peer performance analysis
 - There may be requirements of a few graphs
- **Middle-level Management**
 - Interested in analyzing the performance of different units of administration
 - Interested in peer performance analysis
 - Need for drill down and exception reports
 - Charts for comparing and contrasting performances against National, State and district performances

- **Senior Management**
 - needs revolve around identifying outliers, evaluating the performance of the system so that it can be used as an aid to improve the system through new/revised policies and guidelines
 - Monitoring overall performance across various levels of Government, trend analysis etc.
 - Interested in identifying weaknesses and loopholes in the system so that they can be plugged through improved policies and guidelines
 - Has little or no interest in individual unit/person's performance
- **General Public**
 - A normal citizen may be interested in details of money spent on a particular work being taken up in his locality, proposed plans etc.
 - A more informed citizen/NGO may be interested in knowing more details such as what are the different sources of funding for the municipality, what are the different projects/activities taken up by the government,
 - Need data in computable format
 - Need clarity/explanation on the various parameters being used to filter or report, the relationship between various parameters, the overall purpose of the report etc.

5.4

Setting up a Report – The Query Filter

The first step is to allow the user to tailor the contents of the report. This is usually done through a query filter page. Following may be observed while designing the query filter page:

- a. Based on the expected usage scenario, identify the query parameters which the user would like to configure as per his/her requirements.
- b. One or more parameters may be given to user for specifying values.
- c. If applicable, default values may be provided for some of the query parameters which can be changed as per user requirements
- d. As much as possible, the labels of query parameters should be self-explanatory. In particular, care should be taken to ensure that generic parameters such as Year, Financial

Year, Date etc. clearly indicate exactly what they mean. For eg., if the user is given an option to select “From Date” and To Date”, the user should clearly know which date is being referred to. Ideally speaking, the meaning should be clearly understandable from the label itself.

- e. In addition, help text may be provided against each parameter on hovering of the mouse over the parameter or on clicking a help button provided alongside the parameter.
- f. If such help text is provided, a facility may be given in the top to switch on/off the help as per user’s requirements.
- g. Once the user has chosen the required query parameter values, the parameters will be submitted for generating the report. The generated report may be rendered on the same page or a different page.
- h. If it is proposed to render the report on the same page, then the following care may be taken:
 - i. The query parameters should be submitted using a remote procedure call (eg. AJAX, DWR etc); the advantage of this would be that the entire page need not be rendered again and again
 - ii. Care should be taken to ensure that once the report is rendered based on user-specified criteria, then if the user chooses to change any of the query parameters, then the report should be immediately removed/blanked out. This will ensure that a previous output is not displayed against the newly selected criteria. The report should be regenerated once the new search criteria is submitted.
- i. If it is rendered on a different page, then facility should be given to return back to the query page. When the user returns back to the query page, then ensure that previously selected parameters are shown as selected; this will make it easier for the user to change just one or more parameters and generate a new report.
- j. The query filter page/section should have the following three buttons:
 - Submit button to submit the query for generating the report
 - Reset button to reset the parameter values to their original values (default/blank as the case may be)
 - Close button to close the report
- k. Language option may also be given if there is a requirement to view the report in different languages. In order to render the report in different languages, the following care should be taken:

- The name of a language in the language drop down should be listed in that language
 - The labels and messages should be taken from a property file based on the chosen language
 - All the options in combo boxes, List boxes, Radio Buttons and Check boxes should also be made available in the selected language (which means they should be either taken from a property file or database)
 - If the language is selected, then the report labels should also be displayed in the selected language.
- l. As a precaution against DOS (Denial of Service) attack, some mechanism like CAPTCHA should be introduced to distinguish human from machine input. This is a must to be followed instruction in case the query is very resource centric.
- m. Lastly, the query filter page/section should have a Help button which gives a detailed Help about the report. The help may include
- description of the query parameters,
 - description of the columns that will be generated as part of the report,
 - details of calculations done to generate various columns/rows etc.

A sample format of the query filter page/section is given below for reference:

Logo Report Title Logo

Select ?

Parameter1 : Parameter2 :

Parameter3 : Parameter4 :

Submit Reset Close

5.5

Report Layout

Once the user has configured his requirements in the query parameter section and clicks the Submit button, the report would be generated. This section describes the various components which a report layout should ideally include:

- a. Report Header – Any information to be displayed once at the top of the report. This should include
 - i. the report title,
 - ii. logos (on the left and/right as required)
 - iii. the link to Help page (which was given in the query section of the report. The Help link need not be repeated if the report body is displayed in the same page as the query section)
 - iv. Options to download the report in various formats (Excel, CSV, XML, Word, PDF etc. Option to share the report through email may also be given (in which case the user could be asked to enter the email id to which the report is to be sent)
 - v. Link to print the report
- b. Page Header – Any information to be repeated at the top of each page. The following may be included in each page:
 - i. the Date on which the report was generated
 - ii. A Search facility may be given in the page header to further narrow down the retrieved data set
 - iii. All the query parameters along with the values selected by the user
 - iv. If all columns relating to a particular unit type use the same unit (for eg., if there are multiple columns relating to money and all of them are being displayed in Lakhs of Rs.), then the same may be displayed in the page header.
- c. Report Body – The main body of the report. The main body of the report will invariably consist of a table displaying the data in different columns and/or charts. The following should be observed while displaying the output table:
 - i. Each column should be properly labeled and numbered.
 - ii. As much as possible, use of acronyms in the column titles should be avoided. If it is unavoidable, then a note may be included in the Page Footer explaining the meaning of acronym.

- iii. If domain-specific labels are used as column titles, their meaning may not be apparent to users of reports who are not familiar with the domain. In such cases also, a note may be included in the page footer explaining the meaning of the label.
- iv. The labels should be, as much as possible, self-explanatory. Each column should be clearly explained in the help text. Also, while explaining a particular column, any assumptions made should be clearly indicated and any calculations associated with the column, if any, should also be clearly explained
- v. Use of short forms as values in the report should be avoided as much as possible as it reduces the readability and user friendliness of the report. Such short forms may have been stored in the database and get displayed as it is in the report, either because the concerned officials know what the short form means or it may be that the developer just was too lazy to convert it into user-friendly values. Attempt should be made, as much as possible, to give the complete, user-friendly value. A very common example is the use of the short form “N.A”. It could mean Not Applicable or Not Available. Depending on the context, the wrong interpretation could make a world of difference. If it is not possible to give the complete value (due to space constraints), then a note should be provided in the Page Footer explaining the short forms.
- vi. Another common practice is to leave a value in a column blank; no interpretation can be made of the values shown as blank. If it makes sense, then a default value should be used. The practice of replacing a blank value with zero should be done only if it makes sense to treat it as zero.
- vii. It should be possible to sort the report in ascending and/or descending order of any column. jQuery makes this possible very easily.
- viii. All textual columns should be left-aligned; all numeric (particularly amount) columns should be right aligned. Column names should be Centre-aligned
- d. Page Footer – Any information to be repeated at the bottom of each page. The following should be included in the footer of each page:
 - i. The URL of the application (if it is a web-based application)
 - ii. Page numbers should be displayed in the format “1 of 100” so that the user knows how many pages have been generated as part of the report.
- e. Report Footer – Any information to be displayed at the end of the report
 - i. Website Policy

- ii. The browser and resolution details in which the report can be viewed best

5.5

Emphasize Important Information

Depending on the nature of the report, important information in the report could be highlighted by making it bold. Though color can be used to highlight the information, it can be seen only on screens and in colour printouts. Also, it could be a problem for colour-blind people. Bigger font size, bold etc. could be used. Entire rows or specific cells may be highlighted.

5.7

Format and Paginate

- a. If the body contains more number of rows than that could be displayed in a single screen, efforts should be made to repeat/float the column title to improve the readability.
- b. It is advised to set a cap on the maximum number of records to be fetched while retrieving the data. For eg., If the number of records to be displayed is 350, then the query may fetch records where offset is 1 to 200 only and the control for showing number of records in a single sheet should be limited to 200. Once the user prefers to view next set of records, the query should fetch the records whose offset is 201 to next 200 (in the case explained here it will be 200 to next 150). But while exporting the report to spread sheet or pdf format, the data set may contain complete data.
- c. If required, appropriate water marking may be provided in the body. It may be required if the report is confidential in nature or is a sample report.

5.8

Make the report distributable

Reports may be distributed through various channels:

- a. Through email – The report could be either formatted as a body of the email or attached as a PDF/Excel/CSV/XML file. It will depend on the size of the report. Short reports can be formatted as body but longer reports should ideally be sent as an attachment. The user should be given an option to choose the manner in which the report is to be emailed to the specified email id. The email subject should give the name of the report and the date on which it is generated.

- b. Hosted on the web – This is the normal way in which a report is made available
- c. Print – Printed report is still used in all meetings. It is very important to ensure the following while enabling a report for printing:
 - i. The report should look the same when printed and when viewed online. There should be no need to re-format the report for printing in any way.
 - ii. The online report into pages that fit on an 8.5-by-11-inch piece of paper and check it out by printing a long report to see if it fails any expectations.
 - iii. Colors alone should not contain any information. Reports are printed in black ink, so any color code should be captured in tonal variations.

5.9

Design Database specifically for Reports

As mentioned in the introduction, when we begin to design a software application, our focus is primarily on input screens and the associated tables where the entered data will be stored. However, the main purpose of building any application is to generate meaningful reports for the client department so that they can monitor and take informed decisions. The management primarily concentrates on what the stored data can reveal about their programme or performance of an entity and how they can use that information to improve the performance so that the overall governance objective is effectively met. In this section, we will discuss the differing requirements of data entry and report generation and focus on design strategies we can adopt to address these differences.

5.9.1 TRANSACTIONAL VS REPORTING DATABASE

There are two major type of requirements of any IT system: Data entry and reporting. A data entry system is generally called a Transactional or On-line Transaction Processing (OLTP) System. A reporting system whose primary focus is to generate reports is invariably called an analytic or On-line Analytical Processing System.

A Transactional system or the OLTP system is mainly concerned with a large number of transactions involving Insert, Update and Delete operations. A transactional system is the one which is used by users to manage data entry operations. The database design that is followed is usually the 3NF (the Third Normal Form).

An analytical system or the OLAP system is, on the other hand, characterized by low volume of transactions. Instead, they involve very complex queries and aggregations carried out on large data for generating various reports. Performance is an effectiveness measure of such systems and invariably data needs to be stored in a de-normalized fashion in multi-dimensional schemas (star schema).

Invariably, we end up using the transactional database for reporting system as well. This creates problems both for transactions as well as for the reports as the two systems have conflicting requirements. The conflicting requirements of the two types of systems are listed below in the following table:

Requirements	Transactional (OLTP) System	Analytical (OLAP) System
Type of Data	Deals with operational data; is the original source of data	Historical and Aggregated data; the OLTP is the source of this data
Purpose	To capture data related to operational activities	To help with planning, problem solving and decision making
Types of Database Operations	Operations on the database are primarily Insert, Update and Delete. Queries are relatively simple and confined to few records	Operations on the database are primarily Select with little or no inserts, updates and deletes
Space Requirements	Space requirements would be relatively less if the operational data is periodically archived	Usually large as the historical data is required for any type of reporting or analytic requirements; also, there would be a need for creation of pre-build tables
Database Design	Database design should be highly normalized with many tables	Database should be typically de-normalized to reduce complex joins; number of tables is lesser; star and/or snowflake schemas are used
Indexing	Should be minimized as it has impact on inserts and updates	Indexing is critical for effective performance
Backup & Recovery	Operational data is critical; hence backup should be done regularly	Not so critical; data loss can be compensated by simply re-loading the data from OLTP database

As may be seen from the above table, the requirements of the two types of systems are totally different. Hence, it makes good sense to separate the reporting database from the transaction database. But the next question arises: Is there any criteria for deciding when to go for a separate reporting database?

5.9.2 WHEN TO BUILD A SEPARATE REPORTING DATABASE?

The thumb rule is the more critical your transactional system and/or the more sophisticated the reporting system, you must go for splitting the two databases. If your transactional database is for eg., a department's inventory system, you need not go for a separate reporting database; neither are the transactions so mission critical nor or any sophisticated analysis done on such a system. However, if it is a large system where a large number of users are entering data in real-time, then it may not be a good idea to run complex and/or long running queries on such a database to generate reports as this would adversely impact the availability of the system for data entry. Also, if there are variety of complex reports which require strong indexing and de-normalization to reduce complicated joins, then again it is a fit case to go for a separate reporting database. Thus, small systems with requirements for simple reports need not go for separate reporting database.

5.9.3 ADVANTAGES OF HAVING A SEPARATE REPORTING DATABASE

There are many advantages to maintaining a separate reporting database:

- a. The structure of the reporting database can be specifically designed to make it easier to write reports.
- b. You don't need to normalize a reporting database, because it's read-only. Feel free to duplicate data as much as needed to make queries and reporting easier.
- c. The development team can refactor the operational database without needing to change the reporting database.
- d. Queries running against the reporting database don't add to the load on the operational database.
- e. You can store derived data in the database, making it easier to write reports that use the derived data without having to introduce a separate set of derivation logic.
- f. You may have multiple reporting databases for different reporting needs.

The only downside to maintaining a separate reporting database is that its data has to be kept up to date. The easiest and most frequently used way of achieving this is by populating it overnight. This often works quite well since many reporting needs work perfectly well with previous day's data.

5.9.4 BEST PRACTICES FOR BUILDING A REPORTING DATABASE

Listed below are a few best practices which should be followed when designing and building a reporting database:

- a. De-normalize the reporting database as much as possible to reduce/eliminate undesirable joins. Feel free to duplicate data as much as needed to make queries and reporting easier.
- b. Use indexing strategies which facilitate quick retrieval of data based on the nature of queries that need to be executed
- c. Create views to support common reporting needs. This will increase re-usability and maintainability.
- d. Extract, Transfer and Load (ETL). The ETL process is a necessary requirement for building any database. The data from the OLTP system needs to be extracted, appropriate transformations applied to the data to meet the needs of the reporting database structure and then load it on the reporting database. Sometimes, further processing of this data may be required; for eg., you may wish to pre-built aggregated tables so that the report can be rendered quickly.
- e. Remove unnecessary data that are not required for reporting purposes

5.10

Reporting Frameworks

Reporting Frameworks provide the much needed facilities to quickly turn out great looking reports. Frameworks are available for Java, .NET and PHP. Frameworks for Java include BIRT, Jasper Report, Pentaho etc. Frameworks for PHP include Reportico PHP Report Designer, PHP Report Maker etc. Microsoft provides the Microsoft Reporting Server.

Most of the reporting frameworks provide some or all of the following capabilities:

- a. Quickly prepare the layout of the report by dragging and dropping required controls
- b. Apply custom CSS on the report
- c. Pre-built templates with pre-defined headers and footers so that uniformity across reports is ensured
- d. Sort data as per the requirements of the report.
- e. Grouping data in a meaningful manner. For eg. You may want to group data state or district wise or according to some other parameter specific to the report

- f. Dataset for a report can be from multiple data sources such as CSV, excel, different RDBMS, web service etc.
- g. Output reports in HTML, XML, CSV, JSON, PDF etc
- h. Filtering Data – You can easily add customizable parameters to a report (e.g. start date and end date) for filtering the data that is retrieved from the data source. The facility to select the parameter values can be given to the end user.
- i. Add graphs and charts
- j. Attach Javascripts to make the reports interactive
- k. Presenting Data in Cross Tabs – You can easily configure the framework to display the data in a crosstab format. A cross tab displays data in a row-column matrix by grouping values by one set of data listed down the left side of the matrix and another set of data listed across the top of the matrix.
- l. Designing multi-page reports – Most reports display on multiple pages. You can easily control pagination by specifying where page breaks occur and page numbers, report titles etc. should be displayed.
- m. Creating drill-down reports
- n. Adding interactive viewing features – Some frameworks give additional interactive features such as adding hyperlinks, bookmarks, table of contents etc.
- o. Localizing text –Many times, it is required to display the report in state-specific languages. Some of the frameworks give facility to localize text by displaying labels from resource bundles

5.11

Conclusion

Reports are the soul of any information system. Reports enable the managers and decision makers to evaluate the processes and take informed decisions. It acts as a feedback mechanism to improve the system. More and richer the reports, better is the ability of the decision maker to take decisions. Reporting Frameworks available in different platforms make it all the more easier to build user-friendly reports in a very short time.

- 6.1 FRAMEWORK DESIGN PATTERN**
 - 6.1.1 INVERSION OF CONTROL**
 - 6.1.2 EXTENSIBILITY**
 - 6.1.3 NON-MODIFIABLE FRAMEWORK CODE**
- 6.2 FRAMEWORK COMPONENTS/PARAMETERS**
 - 6.2.1 PACKAGES/WRAPPERS**
 - 6.2.2 ARCHITECTURE**
 - 6.2.2.1 DESIGN PATTERN**
 - 6.2.2.1.1 MODEL-VIEW-CONTROLLER**
 - 6.2.2.1.2 PUSH-BASED VS. PULL-BASED**
 - 6.2.3 METHODOLOGY**
- 6.3 JAVA BASED FRAMEWORKS**
 - 6.3.1 APACHE STRUTS**
 - 6.3.2 SPRING**
 - 6.3.3 JSF**
- 6.4 PHP BASED FRAMEWORK**
 - 6.4.1 MOST POPULAR PHP FRAMEWORKS**
 - 6.4.1.1 LARAVEL**
 - 6.4.1.2 SYMFONY**
 - 6.4.1.3 CODEIGNITER**
 - 6.4.1.4 CAKEPHP**
 - 6.4.1.5 SLIM**
 - 6.4.2 PHP FRAMEWORKS SUMMARY**

APPLICATION DEVELOPMENT FRAMEWORKS

The expectations of government to deliver more services and to deliver them better presents a challenge for NIC. Well-engineered automated solutions can only increase productivity in service delivery to help in meeting these expectations.

The anticipation towards rapid implementation of e-Governance applications has necessitated the improvements in developer productivity along with quality, reliability and robustness of software. The frequent changes in requirements are also need of the hour. It is therefore, essential to focus on the unique requirements of applications instead of spending time on application infrastructure (“plumbing”), so there is a need to have a framework with a set of objects and methods that can be customized/configured for faster delivery of robust applications.

A framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate. A framework is a set of common and prefabricated software building blocks that programmers can use, extend or customize for specific application. With frameworks, developers do not have to start from scratch each time they write an application.

This section provides various alternatives available that can be used as framework to ensure an integrated, coordinated and standards-based effort. In the absence of such a uniting fabric, it is likely that there would be avoidable duplication, incompatibility, delay and inefficiency in delivery of efficient ICT projects.

6.1

Framework Design Pattern

6.1.1 INVERSION OF CONTROL

In a framework, unlike in libraries or in standard user applications, the overall program’s flow of control is not dictated by the caller, but by the framework.

6.1.2 EXTENSIBILITY

A user can extend the framework - usually by selective overriding; or programmers can add specialized user code to provide specific functionality.

6.1.3 NON-MODIFIABLE FRAMEWORK CODE

The framework code, in general, is not supposed to be modified, while accepting user-implemented extensions. In other words, users can extend the framework, but should not modify its code.

6.2

Framework Components/Parameters

6.2.1 PACKAGES/WRAPPERS

A wrapper is way of repackaging a function or set of functions (related or not) to achieve one or more of the following goals (probably incomplete):

- Simplification of use i.e. simplifies an interface to a technology
- Consistency in interface
- Enhancement of core functionality
- Collecting discrete processes into a logical association (an object) i.e. often re-usable regardless of high level design considerations
- Reduces/eliminates repetitive tasks
- Increases application flexibility through abstraction

6.2.2 ARCHITECTURE

Architecture manages a collection of discrete objects and is a style that incorporates specific design elements. Essentially, architecture implements associations between objects--inheritance, container, proxy, collection, etc.

Architectures can and are useful because they create a re-usable structure (a collection of objects) that provides some enhanced functionality, but once you start using them, you're pretty much stuck with them unless you do some major refactoring.

6.2.2.1 Design Pattern

6.2.2.1.1 Model-View-Controller

Model-view-controller (MVC) is a software design pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

6.2.2.1.2 Push-based vs. pull-based

Most MVC frameworks follow a push-based architecture also called “action-based” or “request-based”. These frameworks use actions that do the required processing, and then “push” the data to the view layer to render the results. An alternative to this is pull-based architecture, sometimes also called “component-based”. These frameworks start with the view layer, which can then “pull” results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view.

6.2.3 METHODOLOGY

A methodology enforces the adherence to a consistent design approach, couples object dependencies and are often re-usable regardless application requirements.

6.3

Java based Frameworks

This section deliberates on popular java based frameworks with an objective to provide clear, comprehensive information to ensure rapid development and implementation of applications.

6.3.1 APACHE STRUTS

Apache released Struts Web Application Framework to develop Java EE web tier by considering ease of development and flexibility. The core of the Struts framework is a flexible control layer based on standard technologies like Java Servlets, Java Beans, Resource Bundles, and XML, as well as various Jakarta Commons packages. Struts encourages application architectures based on the Model 2 approach, a variation of the classic Model-View-Controller (MVC) design paradigm.

6.3.2 SPRING

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework’s core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.

- It utilizes some of the well-known technologies, ORM frameworks, logging frameworks, JEE, JDK timers, Quartz and so on. So, developers don’t have to learn any new technologies or frameworks.
- Inversion of Control (IOC) for managing the beans. This is one of the greatest power for spring projects.
- Spring Boot, you can use spring boot for starting and running your spring applications. You don’t have to worry about any of the configurations.

- Also we have Spring AOP
- Strong community where it is easy to get the help
- All type of configurations are supported : XML, Annotation and Java
- Out of the box support for REST services. This also supports JAX-RS specification.
- Spring 5 going to integrate new React engine to take the latest happening in market. That means road map for the spring mvc is good.

6.3.3 JSF

JavaServer Faces (JSF) is a Java specification for building component-based user interfaces for web applications. It was formalized as a standard through the Java Community Process and is part of the Java Platform, Enterprise Edition. JSF 2 uses Facelets as its default templating system.

The JSF is the standard framework developed by Java Community Process and it is the best WAF for developing Java EE 5 or above web applications.

6.4

PHP Based Framework

Here's what PHP frameworks do:

- Make speed development possible
- Provide well-organized, reusable and maintainable code
- Let you grow over time as web apps running on frameworks are scalable
- Spare you from the worries about low-level security of a site
- Follow the MVC (Model-View-Controller) pattern that ensures the separation of presentation and logic
- Promote modern web development practices such as object-oriented programming tools

6.4.1 MOST POPULAR PHP FRAMEWORKS

6.4.1.1 Laravel

Although Laravel is a relatively new PHP framework (it was released in 2011). Laravel has a huge ecosystem with an instant hosting and deployment platform, and its official website offers many screencast tutorials called Laracasts.

Laravel has many features that make rapid application development possible. Laravel has its

own light-weight templating engine called “Blade”, elegant syntax that facilitates tasks you frequently need to do, such as authentication, sessions, queueing, caching and RESTful routing. Laravel also includes a local development environment called Homestead that is a packaged Vagrant box.

6.4.1.2 Symfony

The components of the Symfony 2 framework are used by many impressive projects such as the Drupal content management system, or the phpBB forum software, but Laravel – the framework listed above – also relies on it. Symfony has a wide developer community and many ardent fans.

Symfony Components are reusable PHP libraries that you can complete different tasks with, such as form creation, object configuration, routing, authentication, templating, and many others. You can install any of the Components with the Composer PHP dependency manager. The website of Symfony has a cool showcase section where you can take a peek at the projects developers accomplished with the help of this handy framework.

6.4.1.3 CodeIgniter

CodeIgniter is a lightweight PHP framework that is almost 10 years old (initially released in 2006). CodeIgniter has a very straightforward installation process that requires only a minimal configuration, so it can save you a lot of hassle. It’s also an ideal choice if you want to avoid PHP version conflict, as it works nicely on almost all shared and dedicated hosting platforms (currently requires only PHP 5.2.4).

CodeIgniter is not strictly based on the MVC development pattern. Using Controller classes is a must, but Models and Views are optional, and you can use your own coding and naming conventions, evidence that CodeIgniter gives great freedom to developers. If you download it, you’ll see it’s only about 2MB, so it’s a lean framework, but it allows you to add third-party plugins if you need more complicated functionalities.

6.4.1.4 CakePHP

CakePHP is already a decade old (the first version was released in 2005), but it’s still among the most popular PHP frameworks, as it has always managed to keep up with time. The latest version, CakePHP 3.0 enhanced session management, improved modularity by decoupling several components, and increased the ability of creating more standalone libraries.

CakePHP has a really remarkable showcase, it powers the websites of big brands such as BMW, Hyundai, and Express. It is an excellent tool for creating web apps that need high-level of security, as it has many built-in security features such as input validation, SQL injection prevention, XSS (cross-site scripting) prevention, CSRF (cross-site request forgery) protection, and many others.

6.4.1.5 Slim

Slim is a PHP micro framework that provides you with everything you need and nothing you don't. Micro frameworks are minimalistic in design, they are excellent for smaller apps where a full-stack framework would be an exaggeration. Slim's creator was inspired by a Ruby micro framework called Sinatra.

Slim is used by many PHP developers for developing RESTful APIs and services. Slim comes with features such as URL routing, client-side HTTP caching, session- and cookie encryption, and it supports "flash" messages across HTTP requests as well. Its User Guide is an easy read, and if you are interested then please visit <https://www.slimframework.com/docs/> for the detailed documentation."

6.4.2 PHP FRAMEWORKS SUMMARY

Framework	PRO's	CON's	PHP Version Required
Laravel	Organize files and code · Rapid application development · MVC architecture (and PHP7) · Unit testing (FAST on HHVM) · Best documentation of any · High level of abstraction · Overloading capabilities using dynamic methods · Tons of out of the box functionality · payment integration with stripe · very strong encryption packages · ORM	· Does NOT work on Shared hosting plans · Does Many queries on your database	5.5.9
Symphony	· High performance, due to byte code caching · Stable · Well documented, maintained, and supported · Very good support and is very mature	· While the documentation is good, there is a steep learning curve. · Companies are moving to MVC Framework architectures and Symfony2 does not support MVC.	5.5.9

Framework	PRO's	CON's	PHP Version Required
CodeIgniter	<ul style="list-style-type: none"> · Very developer friendly Doesn't need any special dependencies or supports · Ability to use normal web hosting services well, using standard databases such as MySQL · Outperforms most other frameworks (non MVC) · Good documentation and LTS (Long Term Support) 	<ul style="list-style-type: none"> · No namespace's, however this can speed up · Not as friendly towards unit testing as others · Few libraries that are built inside the framework 	5.4
CakePHP	<ul style="list-style-type: none"> · Modern framework · Supports PHP 5.5+ · Scaffolding system and Fast builds · Very good for commercial web applications (MIT License) · Database Access, Caching, Validation, Authentication, are built in · Extensive safekeeping tools include cross site · scripting prevention, SQL Injection prevention, · CSRF, and Form Validation · Good Documentation · Actively developed 	<ul style="list-style-type: none"> · Not as good for constructing Restful APIS as Laravel or others listed 	5.5.9
Slim	<ul style="list-style-type: none"> · The fastest RESTful Framework available · Enough documentation to get you off the ground · Perfect for Small rest apis · Actively developed · Add-ons include: HTTP Caching, & Flash 	<ul style="list-style-type: none"> · Minimal add-ons on the stock composer when installed. · No official LTS release yet since its very new. 	5.5

APPENDIX

- A. OFFICE ORDERS**
- B. CODE DIRECTORIES**
- C. COMMON DATA ELEMENTS**
- D. PERSON SPECIFIC DATA ELEMENTS**
- E. GEO REFERENCES DATA ELEMENTS**
- F. LABOUR & EMPLOYMENT DATA ELEMENTS**
- G. COMPLIANCE MATRIX**
- H. CASE STUDY FOR COMPLIANCE MATRIX ON DATA QUALITY**
- I. REFERENCES FOR DATA ELEMENTS**

APPENDIX

A.

Office Orders

No. 1(11)/2016-Pers.
Government of India
Ministry of Electronics and Information Technology
National Informatics Centre
A-Block, CGO Complex, Lodhi Road, New Delhi-110003.

Dated : / 9 August , 2016

OFFICE ORDER

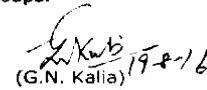
It has been decided with the approval of Director General, NIC to set up a group called "Project Group for Technical Advisory" at NIC Hqrs., New Delhi for development of Enterprise Software Solution in NIC with the following members.

Sl. No.	Name of Officer(s)	Designation	Emp. Code
1.	Shri Girish Kumar Gaur	Scientist-G	2653
2.	Shri Pawan Kumar Joshi	Scientist-F	1104
3.	Shri Rajender Sethi	Scientist-F	287
4.	Shri Jyodeep Shome	Scientist-F	2358
5.	Ms. Alka Mishra	Scientist-F	1975
6.	Ms. Rama Hariharan	Scientist-F	395
7.	Ms. Rachna Srivastava	Scientist-F	2214

2. The overall objective is to develop Enterprise Software Solutions quickly by re-using what is already available in NIC without compromising the quality and reinventing the wheel.

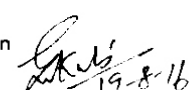
3. The terms of reference will be:

- To help Project Teams, in development of enterprise solution, following common metadata standards, data standards, service oriented architecture, reusable components and already available frameworks.
- The group will advise NIC project team leads so that they can guide and arrange appropriate know-how required by the developers before developers plunge into coding.
- The groups will also advise NIC project team leads to ensure that SDLC processes are followed to the extent possible by taking services from other NIC Groups.


(G.N. Kalra)
Joint Director (Pers.)

Copy to:

- Pay and Accounts Officer, NIC Hqrs., New Delhi
- DDO, NIC Hqrs., New Delhi
- OSD to DG, NICfor information
- Shri Girish Kumar Gaur, Scientist-G
- Shri Pawan Kumar Joshi, Scientist-F
- Shri Rajender Sethi, Scientist-F
- Shri Jyodeep Shome, Scientist-F
- Ms. Alka Mishra, Scientist-F
- Ms. Rama Hariharan, Scientist-F
- Ms. Rachna Srivastava, Scientist-F
- All DDGs/HoGs/SIOs/SDTCs Incharge.....through IntraNIC
- All Sections / Divisions, NIC Hqrs., New Delhithrough IntraNIC
- DDG (Pers. & Admin.), NIC Hqrs., New Delhi
- Director, Vigilance Unit, Deity / VO NIC for information
- HOD, eOffice / OAD / NIC MAIL, NIC Hqrs., New Delhi..... for necessary updation
- Guard File/Personal File/Manpower File/Notice Board


(G.N. Kalra)
Joint Director (Pers.)

No. 1(11)/2016-Pers.
Government of India
Ministry of Electronics & Information Technology
National Informatics Centre
A-Block, CGO Complex, Lodhi Road, New Delhi-110003

Dated: 24th April, 2017

OFFICE ORDER

Subject: Re-constitution of "Project Group for Technical Advisory".

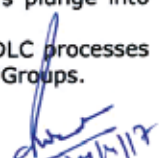
The undersigned is directed to refer to Office Order of even number dated 19.08.2016 and to convey the approval of Director General, NIC for re-constitution of the "Project Group for Technical Advisory" for development of Enterprise Software Solution in NIC with the following composition:

Sl. No.	Name of Officer(s)	Designation	Emp. Code
1.	Dr. (Ms.) Savita Dawar	Scientist-F	2566
2.	Shri Pawan Kumar Joshi	Scientist-F	1104
3.	Shri Rajender Sethi	Scientist-F	287
4.	Shri Joydeep Shome	Scientist-F	2358
5.	Ms. Alka Mishra	Scientist-F	1975
6.	Ms. Rama Hariharan	Scientist-F	395
7.	Ms. Ranchna Srivastava	Scientist-F	2214
8.	Dr. Rajesh Kumar Mishra	Scientist-E	3231 ... Convener

2. The overall objective is to develop Enterprise Software Solutions quickly by re-using what is already available in NIC without compromising the quality and reinventing the wheel.


3. The terms of reference will be as follows:

- To help Project Teams, in development of enterprise solution, following common metadata standards, data standards, service oriented architecture, reusable components and already available frameworks.
- The group will advise NIC project team leads so that they can guide and arrange appropriate know-how required by the developers before developers plunge into coding.
- The group will also advise NIC project team leads to ensure that SDLC processes are followed to the extent possible by taking services from other NIC Groups.


(O.P. Wadhwa)
Joint Director (Pers.)

Copy to:

- Pay and Accounts Officer, NIC Hqrs., New Delhi
- DDO, NIC Hqrs., New Delhi
- OSD to DG, NIC ... for information
- All Members of the Project Group for Technical Advisory
- Shri Girish Kumar Gaur, Scientist-G
- All DDGs/HOGs/SIOs/SDTCs Incharge....through IntraNIC
- All Sections/Divisions, NIC Hqrs., New Delhi
- DDG (Pers.)/DDG(Admn.), NIC Hqrs., New Delhi
- Director, Vigilance Unit, MeitY... for information
- HOD, OAD, NIC Hqrs., New Delhi ... for updation
- Guard File/Personal File/Notice Board through IntraNIC


(O.P. Wadhwa)
Joint Director (Pers.)

B.

Code Directories**1. COUNTRY CODE**

S. No.	Attribute	Attribute Value/Description
1	Name	country_code
2	Aliases	Country
3	Description	Unique country code
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Char
8	Data Length	3
9	Acceptable Values	
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	3 digits including leading zeros
13	Verification	
14	Data Availability	LGD
15	Data Input	Combo box; Country Names to be displayed in alphabetical order
16	Field Caption	Country
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 CD02.01 and ISO 3166 - Country Codes

2. STATE CODE

S. No.	Attribute	Attribute Value/Description
1	Name	state_code
2	Aliases	State
3	Description	Unique State code
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Char
8	Data Length	2
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	2 digits including leading zeros
13	Verification	
14	Data Availability	For details about the service, pl. refer the document available at http://lgdirectory.gov.in/webService/documentation
15	Data Input	Combo box; Names to be displayed in alphabetical order
16	Field Caption	State
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 CD02.02

3. DISTRICT CODE

S. No.	Attribute	Attribute Value/Description
1	Name	district_code
2	Aliases	District
3	Description	Unique code of the districts of India
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Char
8	Data Length	3
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	3 digits including leading zeros; The code of the selected district should be available in the code directory.
13	Verification	
14	Data Availability	For details about the service, pl. refer the document available at http://lgdirectory.gov.in/webservice/documentation
15	Data Input	Combo box; Names to be displayed in alphabetical order; Display only those districts which belong to the State if a State has been selected;
16	Field Caption	District
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 CD02.03

4. SUB DISTRICT

S. No.	Attribute	Attribute Value/Description
1	Name	subdistrict_code
2	Aliases	
3	Description	Sub Districts are administrative units below the level of districts. States use varying names for their sub-districts such as mandal, circle, Tehsil, Taluka etc.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	5
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	Five digits including leading zeros; The code of the selected sub-district should be available in the code directory; If the sub-district of a specific district in a State has to be selected, then a validation should be done at the server level to ensure that the code of the selected sub-district belongs to the selected district of the selected state.
13	Verification	
14	Data Availability	For details about the service, pl. refer the document available at http://lgdirectory.gov.in/web/service/documentation
15	Data Input	Combo box; Names to be displayed in alphabetical order; Display only those sub-districts which belong to the District if a District has been selected;
16	Field Caption	Sub District
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 CD02.04

5. VILLAGE CODE

S. No.	Attribute	Attribute Value/Description
1	Name	village_code
2	Aliases	
3	Description	The unique code of a village
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	6
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	The code of the selected village should be available in the code directory; Based on what parent hierarchy you are following (State>>District>>sub-District>>Village or State>>district>>village or State>>village) to reach to a village, ensure that the code of the selected village falls within the list of villages in the selected parent hierarchy;
13	Verification	
14	Data Availability	For details about the service, pl. refer the document available at http://lgdirectory.gov.in/webservice/documentation
15	Data Input	Combo box; Names to be displayed in alphabetical order; Display only those villages which fall under the parent hierarchy that is used to traverse to the village
16	Field Caption	Village
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 CD02.05

6. LOCAL BODY CODE

S. No.	Attribute	Attribute Value/Description
1	Name	local_body__code
2	Aliases	
3	Description	The unique code of a Local Body. Local Bodies include Urban Local Governments (such as Corporations, Municipalities, Town Panchayats, Notified Area Councils, Cantonment Boards etc.) and Rural Local Governments (such as the three tiers of Panchayats, viz., District Panchayat, Intermediate (block) Panchayat and Gram Panchayat and Traditional Local Bodies established in tribal areas).
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	5
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	The code of the selected local body should be available in the code directory; Based on what parent hierarchy you are following to reach to a local body, ensure that the code of the selected local body falls within the list of local bodies in the selected parent hierarchy;
13	Verification	
14	Data Availability	For details about the service, pl. refer the document available at http://lgdirectory.gov.in/web/service/documentation
15	Data Input	Combo box; Names to be displayed in alphabetical order;
16	Field Caption	Local Body
17	Output Format	
18	Metadata Standards	MDDS Demographic

7. MINISTRY CODE

S. No.	Attribute	Attribute Value/Description
1	Name	ministry_code
2	Aliases	
3	Description	Code of the Central Line Ministries of Government of India
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	2
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	
13	Verification	
14	Data Availability	XML : http://vocab.nic.in/rest.php/orgn/ug/ministry ; JSON : http://vocab.nic.in/rest.php/orgn/ug/ministry/json
15	Data Input	Combo box; Names to be displayed in alphabetical order;
16	Field Caption	Ministry
17	Output Format	
18	Metadata Standards	

8. DEPARTMENT CODE

S. No.	Attribute	Attribute Value/Description
1	Name	department_code
2	Aliases	
3	Description	Unique code of a department of a Central Line Ministry
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	
9	Acceptable Values	As per directory
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	
13	Verification	
14	Data Availability	XML : http://vocab.nic.in/rest.php/orgn/ug/ministry/departments ; JSON : http://vocab.nic.in/rest.php/orgn/ug/ministry/departments/json
15	Data Input	Combo box; Names to be displayed in alphabetical order;
16	Field Caption	Department
17	Output Format	
18	Metadata Standards	

C.

Common Data Elements

1. DATE

S. No.	Attribute	Attribute Value/Description
1	Name	
2	Aliases	
3	Description	These attributes are common for all kind of date data types
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Date
8	Data Length	Not Applicable
9	Acceptable Values	
10	Default Values	Today's date or it may be application specific
11	Mandatory or Optional	Application Specific
12	Validations	Validations, if any, may be done with reference to server date
13	Verification	
14	Data Availability	None
15	Data Input	Date Picker
16	Field Caption	Date
17	Output Format	dd/mm/yyyy
18	Metadata Standards	MDDS Demographic:01 G00.01

2. EMAIL ID

S. No.	Attribute	Attribute Value/Description
1	Name	
2	Aliases	
3	Description	An email address to which an email can be sent.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Varchar
8	Data Length	254
9	Acceptable Values	
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	<p>The format of email address is <local-part>@<domain></p> <p>1. The <local-part> may be up to 64 characters and the entire email address can't be more than 254 characters long.</p> <p>2. The <local-part> format/structure should be - First Five characters are letters, next four numerals, last character letter.</p> <p>The local-part of the email address may use any of these ASCII characters: Uppercase and lowercase Latin letters A to Z and a to z; digits 0 to 9; special characters!#\$%&'*+,-/=/?^_`{ }~; dot .[provided that it is not the first or last character unless quoted, and provided also that it does not appear consecutively unless quoted]</p> <p>The domain name part of an email address must match the requirements for a hostname, a list of dot-separated DNS labels, each label being limited to a length of 63 characters and consisting of</p> <p>uppercase and lowercase Latin letters A to Z and a to z; digits 0 to 9, provided that top-level domain names are not all-numeric; hyphen -, provided that it is not the first or last character.</p>

S. No.	Attribute	Attribute Value/Description
13	Verification	May be verified by (i) sending a verification link to the address with a validity period (ii) sending an OTP; The verified email should not be allowed to change; if required, a confirmation link may be sent to existing email address followed by verification of new email address. In case confirmation link is not responding, the email need to be maintained as Not Verified. A mechanism may be given at an appropriate level to verify such kind of email addresses.
14	Data Availability	None
15	Data Input	Textbox; Preferably 35 characters long with a maximum size of 254 characters.
16	Field Caption	
17	Output Format	The format of email address is <local-part>@<domain>
18	Metadata Standards	MDDS Demographic:01 G00.09

3. MOBILE NO.

S. No.	Attribute	Attribute Value/Description
1	Name	
2	Aliases	
3	Description	Mobile number allocated by a mobile network operator; A mobile number written as +91-XXX YYY ZZZZ is valid throughout India (91 is the country code for India), and in other countries where the + is recognized as a prefix to the country code;
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Varchar
8	Data Length	10 for Indian Mobile Numbers; 14 for International Mobile Numbers
9	Acceptable Values	
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	Indian Mobile Number should be 10 digits long; first digit should be greater than or equal to 7;
13	Verification	May be verified by (i) sending a sending an OTP; (ii) getting a missed call. The verified mobile should not be allowed to change; if required, a confirmation sms may be sent to existing OTP followed by OTP verification of new mobile number. In case confirmation OTP is not responding, the mobile need to be maintained as Not Verified. A mechanism may be given at an appropriate level to verify such kind of mobile numbers.
14	Data Availability	None
15	Data Input	Textbox
16	Field Caption	
17	Output Format	For Indian mobile +91 <Number>
18	Metadata Standards	MDDS Demographic: 01 G00.06-02-05

4. LANDLINE NO

S. No.	Attribute	Attribute Value/Description
1	Name	
2	Aliases	
3	Description	Landline or phone number is the number of a phone which is connected to a fixed telephone line. The format would vary based on whether Indian land line numbers are being captured or international landline numbers are being captured. The format of both is 1. Indian land line numbers are at most 8 digits long without area/STD Code. The total length of all landline numbers (including area/STD code and the phone number) in India is constant at 11 digits; International landline number format of a land line number is +<country code>-<area code>-phone number
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Varchar
8	Data Length	Maximum of 8 digits for subscriber number only; 11 digits for area/STD code and subscriber number; 14 for international number
9	Acceptable Values	
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	All national land phone numbers should be combined with area code and should start with 0; The land phone field can accept digits (0-9), + and – only.
13	Verification	May be verified by getting a missed call. The verified phone should not be allowed to change; if required, a mechanism may be given at an appropriate level to verify such kind of mobile numbers.
14	Data Availability	None
15	Data Input	Textbox
16	Field Caption	
17	Output Format	If country code needs to be prefixed, the format should be +<country code>-<area code>-phone number
18	Metadata Standards	MDDS Demographic:01 G00.06-00-01

5. FINANCIAL YEAR

S. No.	Attribute	Attribute Value/Description
1	Name	financial_year
2	Aliases	
3	Description	Financial year is the period between 1 April and 31 March of any year. It is the 12 month period at the end of which account books are closed, profit or loss is computed, and financial reports are prepared for filing.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Char
8	Data Length	9
9	Acceptable Values	
10	Default Values	Current Financial Year
11	Mandatory or Optional	Application Specific
12	Validations	First digit can't be 0.; Two parts separated by a – (hyphen); The first part contain four digits; The second part should be one number greater than the first part.
13	Verification	
14	Data Availability	None
15	Data Input	Combo box; Textbox - Size 9 Characters; Only digits and one hyphen at the centre are allowed.
16	Field Caption	Financial Year
17	Output Format	YYYY-YYYY (eg.1987-1988)
18	Metadata Standards	

6. PIN CODE

S. No.	Attribute	Attribute Value/Description
1	Name	Pincode
2	Aliases	Pincode, Postal Index Number, PIN
3	Description	A Postal Index Number or PIN or Pincode is a code in the post office numbering or post code system used by India Post, the Indian postal administration. The code is six digits long.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Char
8	Data Length	6
9	Acceptable Values	
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	All digits; First digit can't be 0.;
13	Verification	
14	Data Availability	List of Pin codes are available at https://data.gov.in/catalog/all-india-pincode-directory
15	Data Input	Textbox
16	Field Caption	Pincode
17	Output Format	999999
18	Metadata Standards	MDDS Demographic:01 G02.04-01

7. IFSC CODE

S. No.	Attribute	Attribute Value/Description
1	Name	bank_ifsc
2	Aliases	IFSC Code
3	Description	The Indian Financial System Code (also known as IFSC) is an alphanumeric code that facilitates electronic funds transfer in India. A code uniquely identifies each bank branch participating in the two main Payment and settlement systems in India: NEFT and RTGS.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Char
8	Data Length	11
9	Acceptable Values	
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	First four alphabets, 5th zero; last six characters alphanumeric
13	Verification	
14	Data Availability	The complete list is available with Reserve Bank of India. As on the date of preparing this document, no complete list is exposed by RBI in public domain.
15	Data Input	Textbox (11 Characters)
16	Field Caption	Bank IFSC
17	Output Format	AAAA0999999
18	Metadata Standards	

D.**Person Specific Data Elements****1. AADHAAR NUMBER**

S. No.	Attribute	Attribute Value/Description
1	Name	aadhaar_number
2	Aliases	Aadhaar, UID
3	Description	The Unique code given to residents by the UIDAI. Pl. Note the use of the word “Residents”. Aadhaar cannot be used as a proof of citizenship
4	Data Source	Aadhaar Card
5	Base or Derived	Base
6	Privacy and Security	Private
7	Data Type	Integer
8	Data Length	12
9	Acceptable Values	Though Aadhaar numbers are available as part of UIDAI’s repository, they are not available as a list of values
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Must be exactly 12 digits; No leading zeros; Check the format/structure using Verhoeff Algorithm; Other validation rules may be applied as is applicable to the domain for e.g. Aadhaar number should not be duplicate
13	Verification	(i) Off-line using QR Code on Aadhaar Card (ii) On-line using e-Auth (Aadhaar Number and Name) (iii) On-line using e-KYC i.e. Biometric Authentication – Aadhaar number along with either Fingerprint or Iris Biometrics (iv) On-line using OTP – Using Aadhaar number to send OTP to the registered mobile number (v) More than one type of authentication (multi-factor authentication) may be used based on application requirement

APPENDIX

S. No.	Attribute	Attribute Value/Description
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Aadhaar Number
17	Output Format	9999 9999 9999 or masked as xxxx xxxx 9999
18	Metadata Standards	MDDS Demographic:01 G01.01

2. PAN NUMBER

S. No.	Attribute	Attribute Value/Description
1	Name	pan
2	Aliases	Permanent Account Number
3	Description	Permanent Account Number (PAN) is issued by the Income Tax Department, to any “person” who applies for it or to whom the department allots the number without application.
4	Data Source	PAN Card
5	Base or Derived	Base
6	Privacy and Security	Private
7	Data Type	Character
8	Data Length	10
9	Acceptable Values	As per format
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Must be exactly 10 alphanumeric characters; The first five characters are alphabets, next four numerals, last character alphabets; Other validations as per application requirement
13	Verification	(i) API based on-line PAN Verification from NSDL
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Permanent Account Number or PAN
17	Output Format	AAAAA9999A or masked as xxxxx 9999A
18	Metadata Standards	

3. FULL NAME (IN ENGLISH)

S. No.	Attribute	Attribute Value/Description
1	Name	full_name
2	Aliases	Name, Full Name
3	Description	Full name of a person in English
4	Data Source	Any document like form, register etc
5	Base or Derived	Base
6	Privacy and Security	
7	Data Type	Varchar
8	Data Length	99
9	Acceptable Values	As per format
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Must contain alphabets, space, dot only; Should not be left blank
13	Verification	Manual
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Name
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 G01.02-02

4. GENDER

S. No.	Attribute	Attribute Value/Description
1	Name	gender
2	Aliases	
3	Description	Gender of a person
4	Data Source	Any document like form, register etc
5	Base or Derived	Base
6	Privacy and Security	
7	Data Type	Char
8	Data Length	1
9	Acceptable Values	M-Male; F-Female; T-Transgender
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Acceptable Values only
13	Verification	Manual
14	Data Availability	None
15	Data Input	Combo Box or Radio Buttons
16	Field Caption	Gender
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 G01.03

5. MARITAL STATUS

S. No.	Attribute	Attribute Value/Description
1	Name	marital_status
2	Aliases	
3	Description	Marital status of a person
4	Data Source	Any document like form, register etc
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	
9	Acceptable Values	1-Never Married; 2-Married; 3-Widow/Widower; 4-Divorced; 5-Separated
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Acceptable Values only
13	Verification	
14	Data Availability	None
15	Data Input	Combo Box or Radio Buttons
16	Field Caption	Marital Status
17	Output Format	
18	Metadata Standards	MDDS Demographic:01 G01.04

6. DATE OF BIRTH

S. No.	Attribute	Attribute Value/Description
1	Name	date_birth
2	Aliases	DOB, Birth Date, Date of Birth
3	Description	Date of birth of a person
4	Data Source	Any document like form, register, birth certificate etc
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Date
8	Data Length	Not Applicable
9	Acceptable Values	
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Should always be less than or equal to today's date (server date)
13	Verification	Manual
14	Data Availability	None
15	Data Input	Date Picker
16	Field Caption	Date of Birth
17	Output Format	dd/mm/yyyy
18	Metadata Standards	MDDS Demographic:01 G00.01

7. DATE OF BIRTH TYPE

S. No.	Attribute	Attribute Value/Description
1	Name	date_birth_type
2	Aliases	DOB Type, Birth Date Type
3	Description	This would be mainly used to flag , if the Date of Birth is actual or declared, in case, the Person is not sure about Date of Birth
4	Data Source	Any document like form, register, birth certificate etc
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Character
8	Data Length	1
9	Acceptable Values	D-Declared; V-Verified; A-Approximate (calculated from Age on the date of filling form)
10	Default Values	D-Declared
11	Mandatory or Optional	Application Specific
12	Validations	Acceptable values only
13	Verification	
14	Data Availability	None
15	Data Input	Combo Box, Radio Button
16	Field Caption	Date of Birth Type
17	Output Format	Text
18	Metadata Standards	MDDS Demographic:01 G01.16

8. APPELLATION CODE

S. No.	Attribute	Attribute Value/Description
1	Name	appellation
2	Aliases	Appellation, Title, Prefix to Name
3	Description	An Appellation is title for a person to be prefixed with his/her name. It represents:
4	Data Source	An Appellation is title for a person to be prefixed with his/her name. It represents: a. Gender & Marital status like Mr., Mrs., Ms. b. Attained professional educational qualification like Dr., CA, Engineer etc. A Person can have maximum two Appellations as prefix to person name.
5	Base or Derived	Base
6	Privacy and Security	Application Specific
7	Data Type	Integer
8	Data Length	2
9	Acceptable Values	For Appellation codes & Values refer to Code Directory (MDDS Demographic:01 CD01.04)
10	Default Values	
11	Mandatory or Optional	Application Specific
12	Validations	Acceptable values only
13	Verification	
14	Data Availability	None
15	Data Input	Combo Box for selection of maximum two appellations (one based on profession and other based on gender/ marital status)
16	Field Caption	Title (Prefix)
17	Output Format	The sequence of display / printing of multiple applications would be as follows: Attained professional appellation, followed by the “Gender & Marital status” appellation within brackets.
18	Metadata Standards	MDDS Demographic:01 G01.05-01

E.

Geo References Data Elements

1. LONGITUDE

S. No.	Attribute	Attribute Value/Description
1	Name	longitude
2	Aliases	Longitude
3	Description	X co-ordinate of geographic co-ordinate system (GCS) for locations. Longitudes are the vertical circles on the globe showing the measurement of object's location vertically on the earth's surface in terms of radians measured from the centre of the earth. The dimension of all meridians of longitudes is same and they converge with each other on the North and South Pole making a point of interaction on the earth's surface on the North and South Pole. It is also measured in terms of Degrees, Minutes and Seconds. Note: The dimension of equator and meridians of longitude is same.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Private
7	Data Type	Numeric
8	Data Length	
9	Acceptable Values	Decimal values +/-0-180
10	Default Values	None
11	Mandatory or Optional	Mandatory
12	Validations	Must have at least 5 decimal places
13	Verification	
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Longitude
17	Output Format	+/- 999.99999 (999d 99m 99.99s)
18	Metadata Standards	MDDS- Demographic:01 G02.05-00-01 (To be taken up)

2. LATITUDE

S. No.	Attribute	Attribute Value/Description
1	Name	latitude
2	Aliases	Latitude
3	Description	Y co-ordinate of geographic co-ordinate system (GCS) for locations. Latitudes are the horizontal circles on the globe showing the measurement of object's location horizontally on the earth's surface in terms of radians measured from the centre of the earth. The biggest parallel of latitude on the earth's surface is equator and the parallels of latitude dimension decreases gradually from the equator towards the North and South Pole. It is measured in terms of Degrees, Minutes and Seconds.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Private
7	Data Type	Numeric
8	Data Length	
9	Acceptable Values	Decimal values +/-0-90
10	Default Values	None
11	Mandatory or Optional	Mandatory
12	Validations	Must have at least 5 decimal places
13	Verification	
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Latitude
17	Output Format	+/- 99.99999 (99d 99m 99.99s)
18	Metadata Standards	MDDS- Demographic :01 G02.05-00-02 (To be taken up)

3. ALTITUDE

S. No.	Attribute	Attribute Value/Description
1	Name	altitude
2	Aliases	Altitude
3	Description	A measure of distance above mean sea level. Z co-ordinate of geographic co-ordinate system (GCS) for locations.
4	Data Source	
5	Base or Derived	Base
6	Privacy and Security	Private
7	Data Type	Numeric
8	Data Length	
9	Acceptable Values	Decimal values +/-
10	Default Values	None
11	Mandatory or Optional	Optional
12	Validations	
13	Verification	
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Altitude
17	Output Format	+/- 99999.99 metres
18	Metadata Standards	MDDS- Demographic:01 G02.05-00-03 (To be taken up)

F.

Labour & Employment Data Elements**1. LABOUR IDENTIFICATION NUMBER (LIN)**

S. No.	Attribute	Attribute Value/Description
1	Name	labour_identification_number
2	Aliases	Labour Identification Number, LIN, Shram Pechaan Sankhya
3	Description	LIN is issued by Ministry of Labour & Employment to establishments covered under any of the labour laws.
4	Data Source	Shram Suvidha Portal
5	Base or Derived	Base
6	Privacy and Security	
7	Data Type	Character
8	Data Length	10
9	Acceptable Values	
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Must be exactly 10 digits; No leading zeros; Check the format/structure using Verhoeff Algorithm; Other validation rules may be applied as is applicable to the domain e.g. LIN should not be duplicate
13	Verification	On-line using web service currently available to EPFO/ESIC only.
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	Labour Identification Number
17	Output Format	9-9999-9999-9
18	Metadata Standards	

2. NATIONAL CLASSIFICATION OF OCCUPATIONS (NCO)

S. No.	Attribute	Attribute Value/Description
1	Name	nco
2	Aliases	National Classification of Occupation, NCO, Occupation
3	Description	NCO is a classification of occupations which describes and assigns codes to various occupations in the country and aligns it with ISCO.
4	Data Source	National Career Service Portal
5	Base or Derived	Base
6	Privacy and Security	
7	Data Type	Character
8	Data Length	8
9	Acceptable Values	
10	Default Values	None
11	Mandatory or Optional	Application Specific
12	Validations	Must be exactly 8 digits; first digit represents Division (1-9); first three digits – Sub-division (111-999); first four digits – Family (Max 9999); fifth and sixth digit – Occupation; last two digits – QP NOS. Maximum 99 occupations in a family. QP NOS can be 00-99.
13	Verification	
14	Data Availability	None
15	Data Input	Text Box
16	Field Caption	NCO
17	Output Format	9999.9999
18	Metadata Standards	

G.

Compliance Matrix

This appendix identifies those clauses which contain guideline requirements that need to be addressed during the course of development of e-governance applications. Following checklists may be used to find the level of compliance against these guidelines.

1. Name of the Project: _____
2. Naming Convention Used: _____
3. Data Quality

S. No.	Guideline	Reference	Total Elements	Compliance Elements Covered	Level (1-10)
1	2	3	4	5	6
A	Data Elements	2.1			
1	Identification	2.1.1			
2	Verification	2.1.5			
3	Availability	2.1.6			
4	User Interface	2.1.7			
5	Metadata Standards	2.1.8			
B	Record Element	2.2			
C	Data Function	2.3			
D	Identifiers	2.4			

4. AUTHENTICATION

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
1	2	3	4	5	6	7	8	9
1	Implementation of Authentication	3.3						
2	Sign-up/login Processes	3.4						
3								
...								

5. FORMS

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
1	2	3	4	5	6	7	8	9
1	The user interface (appearance and behaviour) is standardised across application	4.1						
2	The Title/name of the form is properly defined and is based on purpose of the form	4.1.1						
3	The form is checked for normal flow by asking questions/data in appropriate way. The details are asked in logical order from user's perspective	4.1.1						
4	The terminology/language used for the form is familiar to the user and well understood	4.1.1						
5	The related information is grouped appropriately and each group has a heading	4.1.2						
6	Long forms are broken in sub forms by subject/topic	4.1.2						
7	Field labels on forms clearly explain what entries are desired	4.1.2						
8	Start up page is created for large forms with objectives and instructions	4.1.2						

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
9	Forms pre-warn the user if external information is needed for completion (e.g. PAN Number, Aadhaar Number, passport number etc.)	4.1.2						
10	Labels are placed close to the data entry fields	4.2						
11	Text boxes on forms are the right length for the expected response	4.2						
12	The “required” and “optional” fields on forms are clearly marked	4.2						
13	Fields in data entry screens contain default values when appropriate and show the structure of the data and the field length	4.2						
14	The application automatically formats the entered data in the desired format	4.2						
15	The user is able to Tab through the form using keyboard and Tabs are indexed properly	4.2						
16	Clear visual ‘notification’ is provided by using colour change,highlighting border or fade in box	4.2						

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
17	The Labels are top aligned	4.2						
18	For multi-page forms the progres Indicator is provided to communicate ststus to the user	4.2						
19	For inline Label forms, user is displayed with floating label after entry starts	4.3						
20	The Inline labels can be clearly distinguished from actual data	4.3						
21	Pull-down menus, radio buttons and check boxes are used in preference to text entry fields on forms (i.e. text entry fields are not overused)	4.3.2						
22	Fields on forms contain hints, examples or model answers to demonstrate the expected input	4.3.4						
23	The application is carrying out the field level validation and form level validations at appropriate times	4.4						
24	Both Server Side validation and Client side validations are incorporated appropriately	4.4						

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
25	The business rule validations are incorporated appropriately	4.4						
26	The Security validation are incorporate and form is tested for XSS, SQL injection attacks	5.4						
27	The validation feedback is properly incorporated at appropriate place in red color with icon	4.4.3						
28	The errors are properly communicated with instruction for fixing them	4.4.3						
29	For file upload Restriction on Document type and extension is incorporated in validation	4.5.1.3						
30	Restriction on file size is incorporated as per requirement	4.5.1.3						
31	Proper file format is selected as per context of its use	4.5.1.3						
32	The security vulnerabilities are addressed and tested properly for file upload	4.5.1.4						

6. REPORTS

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
1	2	3	4	5	6	7	8	9
1	Report Design	5.3						
2	Reporting Frameworks	5.4						
3								
...								

7. FRAMEWORKS

S. No.	Guideline	Section Reference	Applicability		Conformance			
			Yes/No	If no, reason	Yes	Partial	No	Comments
1	2	3	4	5	6	7	8	9
1	Java based Frameworks	6.3						
2	PHP Based Framework	6.4						
3								
...								

H.

Case Study for Compliance Matrix on Data Quality

1. Project Name: Employee Data Management
2. Naming Convention Used: Lower Case Embedded Underscore

DATA DICTIONARY

S. No.	Data Element	Data Type	Size	Identification	Verification	Availability	User Interface	Standards
1	employee_id	number		Y	NA	NA	Y	NA
2	full_name	varchar	99	Y	Y	NA	Y	Y
3	gender	character	1	Y	Y	Y	Y	Y
4	date_of_birth	date		Y	Y	NA	N	Y
5	date_birth_type	character	1	Y	NA	Y	Y	Y
6	age	integer		Y	N	NA	Y	NA
7	premise_no_or_name	varchar	40	N	NA	NA	Y	Y
8	sub_locality_or_colony	varchar	40	Y	NA	NA	Y	Y
9	locality_or_village	varchar	40	Y	NA	NA	Y	Y
10	district	character	3	Y	NA	Y	Y	Y
11	pincode	character	6	Y	NA	NA	Y	Y

S. No.	Data Element	Data Type	Size	Identification	Verification	Availability	User Interface	Standards
12	state	character	2	Y	NA	N	Y	N
13	PAN	character	10	N	Y	NA	Y	Y
14	aadhar	number	12	N	Y	NA	Y	Y
15	retirement_date	date		Y	NA	NA	Y	Y
16	board_university	varchar	99	Y	NA	NA	Y	NA
17	exam_passed	varchar	50	Y	NA	NA	Y	NA
18	exam_passing_year	integer		Y	NA	NA	Y	NA
19	department_name	varchar	99	Y	NA	NA	Y	NA
20	designation	varchar	25	Y	NA	NA	Y	NA
21	joining_date	Date		Y	NA	NA	Y	Y
22	rel_date	Date		N	NA	NA	Y	Y

The assumptions for compliance are as follows

1. There are 22 data element in the dictionary.
2. The employee-id uniquely identifies an employee with first two digits represents department code in the employee id followed by running serial number. The Employee Id will be unique identifier created based on the criteria defined in Identifier Section except semantic-free criteria. So accordingly, compliance level has been kept below 10.
3. There are three data functions namely employee_master, employee_Qualification, emp_exp. The data structure for these

has all the elements as per record identification, but does not contain last updated/verified elements. Due to incomplete identification, the compliance level again is kept below 10.

4. The record elements does have creation date but not have last updated by and on which date.
5. There are 15 data elements which are defined in the standards, only 14 have been referred. It is assumed that all the 14 elements follow metadata standards.
6. The compliance level refers to degree of compliance in various forms/interface. It possible that a data element may have user interface as per guidelines and in some forms/reports, it may not meet all the UI requirements. Accordingly, compliance level may be assessed.
7. There are 4 data elements used in the project for which data is available, but only for 3 data elements, the same has been used.
8. Regarding data verification, there are 6 elements which are verifiable, but the verification mechanism has been used for 5 data element. Further, depending on the verification interface, the compliance level can be decided.
9. Out of 22 data elements, either ambiguity in name or the naming convention as decided for project has not been used for PAN, rel_date, premise_no_or_name. The spelling of Aadhaar is wrong.

APPENDIX

In view of the above assumptions, one can arrive at a value indicating the level of compliance in terms of percentage. One can say, the above project is 78.29% compliant with respect to Data Quality Guidelines as per compliance matrix given below:

S. No.	Guideline	Reference	Total Elements	Compliance		Score		
				Elements Covered	Level (1-10)	Compliance	Max	%age
1	2	3	4	5	6	7 = 5x6	8 = 4x(10)	9
A	Data Elements	2	22					
1	Identification	2.1	22	18	10	180	220	81.82%
2	Verification	2.5	6	5	9	45	60	75.00%
3	Availability	2.6	4	3	9	27	40	67.50%
4	User Interface	2.7	22	21	7	147	220	66.82%
5	Metadata Standards	2.8	15	14	10	140	150	93.33%
B	Record Element	3	3	3	7	21	30	70.00%
C	Data Function	4	3	3	9	27	30	90.00%
D	Identifiers	5	1	1	8	8	10	80.00%
Total Score						595	760	78.29%

I.

References

- NIC Office Order No. 1(11)/2016-Pers. Dated 19th August 2016 regarding setting-up of Project Group on Technical Advisory.
- NIC Office Order No. 1(11)/2016-Pers. Dated 24th April 2017 regarding Re-constitution of Project Group for Technical Advisory.
- The Open Web Application Security Project (OWASP) on Data Validations https://www.owasp.org/index.php/Data_Validation
- The Open Web Application Security Project (OWASP) on Authentication Cheat Sheet https://www.owasp.org/index.php/Authentication_Cheat_Sheet
- Standards for e-Governance Applications (<http://egovstandards.gov.in>)
- Aadhaar Authentication API specification https://uidai.gov.in/images/FrontPageUpdates/aadhaar_authentication_api_1_6.pdf
- UID Numbering Scheme https://uidai.gov.in/images/uid_numbering_scheme_1.pdf
- Local Government Directory (<http://lgdirectory.gov.in>)
- Controlled Vocabulary Service (<http://vocab.nic.in>)
- Improving the User Experience (<http://www.usability.gov/>)
- Web Form Design – Filling the Blanks by Luke Wroblewski
- Forms that Work – Designing Web Forms for Usability by Caroline Jarrett & Gerry Gaffney)
- Best Practices for Web forms (http://static.lukew.com/webforms_lukew.pdf)
- Comparison of Frameworks https://en.wikipedia.org/wiki/Comparison_of_web_frameworks
- Apache Struts <https://struts.apache.org/>
- Apache Wicket <https://wicket.apache.org/>
- Spring <https://spring.io/>
- Java Server Faces <https://jaserverfaces.java.net/>

- Aadhaar https://uidai.gov.in/images/authentication/authentication_standards_and_specs_v1_7.pdf
- Indian Financial System Code (IFSC) <https://www.rbi.org.in/scripts/FAQView.aspx?Id=60>
- Mobile Number <http://www.dot.gov.in/access-services/national-numbering-plan-2003>
- Permanent Account Number (PAN) [http://www.incometaxindia.gov.in/tutorials/1.permanent account number \(pan\).pdf](http://www.incometaxindia.gov.in/tutorials/1.permanent%20account%20number%20(pan).pdf)
- Labour Identification Number (LIN) <https://shramsuvudha.gov.in>
- National Classification of Occupations (NCO) <http://ncs.gov.in>

National Informatics Centre
Ministry of Electronics & Information Technology

"A" Block, CGO Complex,
Lodhi Road, New Delhi 110 003
India

Email: gudapps@nic.in

<http://gudapps.guidelines.gov.in>

ISBN - 978-81-909457-1-4

