

Mobile Application Development Framework



January -2022



**Handbook for Design, Development and Upkeep of Mobile Apps
(For internal use of NIC Officers)**

Table of Contents

Number	Topic	Page
	Foreword	4
1.0	Introduction	5
1.1	Objective	6
1.2	Mobile Device Types and Platforms	7
1.3	Content Organization	8
2.0	App Development Life Cycle	9
2.1	Approaches to App Development	9
2.2	Development Life Cycle	12
2.3	Development Frameworks	13
2.4	Recommended Framework: Flutter	18
2.5	Targeted Users and Purpose	19
2.6	Wireframe of Mobile App	19
3.0	Flutter as Recommended Framework	23
3.1	Setting up Flutter Environment	25
3.2	Demo App Tutorial using SQLITE and APIs	26
3.3	Re-Usable Code Libraries	35
4.0	Web APIs for Communication	41
4.1	RESTful Web Services	43
4.2	API Security: Best Practices	58
4.3	IIS Hosting of Web APIs	61
5.0	Securing Mobile Apps	68
5.1	Developer Challenges	69
5.2	Security Issues - Mobile App Development	69
6.0	App Testing	73
6.1	Cloud-based App Testing	74
6.2	Stages of App Testing	75
6.3	Android Testing Frameworks	82
6.4	iOS Testing Frameworks	83
7.0	Publishing Your App	87
7.1	Publishing on Google Play Store	87
7.2	Recent Changes in Store Policies	94
7.3	Publishing on Apple App Store	95
8.0	Maintenance, Reviews, Enhancements	99
	Contributors	102
	References	103

Versioning

Version	Release Date	Reasons	Content Added/ Updated/ Sections Updated
1.0	20-Sep-2021		Draft content shared with MCCs for feedback
1.1	10-Dec-2021	Review meeting of MCCs by DG NIC on 12-Nov-2021	As per suggestions during the review meeting content of the book has been re-organised and modified as per suggestions. New Chapter on Testing and Contributors /References sections have been added. Chapter 6 Added, Chapters 2 and 3 re-organised. New sections for Contributors and References added. Re-usable code libraries and app security inputs taken from Cyber Security Group added.
2.0	1-Jan-2022	Review meeting of MCCs by DG NIC on 17-Dec-2021	Minor modifications as per suggestions received.

Foreword

Today the focus area in eGovernance, to cover larger user base, is through Mobile Apps because of availability of smart phones with majority of the population in India and extensive mobile connectivity. A forecast of smartphone users in India in 2021 is 85 crores which will make more than 65% population coverage. Therefore, quick development of mobile apps using appropriate technology is definitely the need of hour.

The idea of bringing out this 'Handbook on Mobile Application Development' is a follow up action resulting from the District Governance Mobile Challenge. Almost 646 District Officers of NIC submitted the synopsis and more than 300 mobile apps have been developed during the Challenge period of 2 months. This prompted the need for having NIC's own Framework for Mobile App development. The objective is to enable NIC officers to develop mobile apps to meet immediate Governance needs quickly and, in a user-friendly manner. During the pandemic, smart mobile phones have emerged as the saviours in meeting information requirements of Governments and Citizens alike.

This Framework will help the NIC officers in embarking upon Mobile App development path by following the best practices and using the re-usable code for standard requirements like camera phone, GPS coordinates, image re-sizing, PDF generation etc. The tools to develop apps are suggestive and recommended on the basis of their utility and quickness in producing cross platform apps. Advanced users may opt for tools in which they are comfortable with. The Framework provides best practices and the life-cycle to be followed, especially the hosting requirements, reviews, updation of apps etc. These need to be followed by all developers.

I complement the NIC Mobile Application Division and all Centres of Competency for bringing out this eBook. Suggestions for improvement are always welcome in next editions, which will be required in the ever-changing App scenario and can be shared with Mobile App Division.

Wishing that the eBook is put to Best Use by NIC Officers in providing online services through Mobile Apps.

**Dr. Neeta Verma,
Director General**

A mobile application, commonly referred to as an **App**, is a type of application software designed to run on a mobile device, such as a smart phone, tablet, smart TV, computer or smart watch. Mobile applications are intended to provide users with similar services on touch screen devices as are accessed on PCs using browser. This is achieved with the help of Services/ APIs. Mobile App development can be carried out using various Technologies/Frameworks available today.

A common question raised is **Why Mobile Apps?** The number of smart phone users in India is estimated to be 850 million in 2021, with 73% literacy rate of Indian populace, and further justifications are:

- User convenience, high up-time, easy access, reduced cost, low Carbon Footprints and reduced foot fall in the offices are key factors leading to the Mobile First development. mGovernance is a way forward in the easy delivery of the services. Almost every major citizen of India has a mobile phone (100%, above 15 years of age)
- Internet Connectivity is much cheaper as compared to previous years
- High Bandwidth availability
- Capability to work offline with lesser functionality in case of non-availability of Internet
- One doesn't need to know typing to work on a Smart phone. It only has a touch screen. You just tap and use it.
- You don't need electricity all the time to use a Smart phone. It's always charged, always on, with short duration of charging time.
- Less time taken to develop and add new features in App
- There are no maintenance costs or parts in a smart phone, hence lesser downtime

Therefore, providing citizen interfaces to Government Services through Mobile Apps is a logical reason. NIC being at the forefront of developing eGovernance applications in the country, the need for adding mobile app-based services is an urgent requirement in today's scenario.

NIC officials have developed more than 700 mobile applications and majority of these are accessible from <https://egovmobileapps.nic.in>. Mobile Apps have certain inherent **features**, which make them popular among users, like

- Simplicity- Many people have short attention spans and if you make your app difficult to navigate, then they will lose interest fast.
- Speed- Fast loading screens
- Flexibility, with option to quickly change interface or add new features

- Security
- Easy Search options for content and services within the App
- Bright and bold colour schemes
- Push notifications to alert the user about action to be taken
- User feedback on every services by just tapping the screen

mGovernance and Application Areas

Providing services to different set of users in Government through mobile apps is also known as **mGovernance**. The mobile apps can be useful where services are provided to the citizens and they need to apply through a form and upload documents or photos. The form length is limited and status of submission and service delivery comes as an SMS or mobile notification in the App. The user doesn't have to login repeatedly and remember the User Credentials/ Passwords. It can be used for location-based services like geo-tagging of Government assets, taking feedback, carrying out location-based inspections to ensure that it has been carried out on-location only. Mobile apps are also useful in providing all Government information, contacts, tourist, hospital, police, petrol stations, directions for emergency services, one click based SoS services, voice and face recognition-based services.

1.1	Objective
------------	------------------

With the Smart Mobile phones becoming the first choice of citizens in accessing any kind of resource on Internet, it is imperative that NIC, being the top solution provider for Government Services, provides mobile based G2C, G2B, G2G and G2E solutions. With this target, the District Governance Mobile Challenge was organised by NIC and all District officials participated in this challenge to develop at least one mobile application for their district requirements. Almost all District officers developed one functional mobile app and published it. It not only gave them an idea about mobile application development process and steps involved, but opened a new citizen service medium, MOBILE FIRST.

The methodology adopted to start mobile application development at District level was to arrange a week's online training, with hands on practice/ common code sharing. Resources from 4 Mobile Competency Centres (Chennai, Kannur, Patna and Shimla) imparted these trainings. Thereafter, the queries of District Officers were addressed by respective Mobile Competency Centres and State level teams were constituted for this Challenge to handhold and guide the officers.

While many of the DIOs/ADIOs/DIAs developed very good mobile apps, still there is scope to further improve the quality and scope of the apps. To further strengthen the skill set and knowledge of NIC Officers, it has been suggested to focus more on Mobile Application Development due to the inherent advantages in providing various citizen services and wide mobile reach.

The basic objective of this handbook is to enable the use of suitable technology or framework as per different criteria and requirements so that NIC Officers in the field offices can quickly learn to develop mobile apps for various Government/ Citizens requirements.

For anyone, starting mobile app development, this Handbook provides the complete set of requirements along with resources. As per Mobile App Development Life Cycle, the coverage includes the selection of development tools, wireframe, APIs, local database, security, publishing, re-usable code for specific requirements and App maintenance.

1.2	Mobile Device Types and Platforms
------------	--

There are two dominant platforms in the modern smartphone market. First is the iOS platform from Apple Inc. Secondly, the most popular Platform is Android from Google. The Android operating system is used not only by Google devices but also by many other OEMs to build their own smartphones and other smart devices whereas the iOS (iPhone Operating System) is restricted to only Apple manufactured smart phones/devices. The Apple phones are expensive as compared to Android based smart phones manufactured by different manufacturers.

The Android uses different software development kits (SDKs) and different development toolchain. While Apple uses iOS exclusively for its own devices, Google makes Android available to other companies, provided they meet specific requirements such as including certain Google applications on the devices they ship. Developers can build apps for hundreds of millions of devices by targeting both of these platforms. Earlier there have been Blackberry and Windows OS for smart phones but their numbers declined over the years and hence, Apps are not specifically targeted for these platforms. Windows App, however, have a new Desktop category which works on a Desktop computer just as it would run on a smart phone.



Presently, Android phones cover about 73% of world's smart phone market and iOS (Apple) users are about 26%. However, the trend is different in Indian scenario where almost 95% Android smart phones are in use with only 3.2% iOS users. Therefore, developing apps for the Android OS should be priority but due

to increasing number of iOS users in India, as per trends, it would be advisable to develop apps for both Android and iOS users.

1.3	Content Organization
------------	-----------------------------

The content in this Handbook is organised on the basis of Mobile App Development Life Cycle. The focus is on Cross-platform development using Flutter framework and emphasis is on Flutter, Wireframes for UI/UX testing, APIs, App Security, Local Database, App Publication Guidelines, Maintenance and Reviews. The procedure to download Adobe XD for wireframe and Flutter Framework installation for development purposes has been covered in details. The Web API Communications, use of Flutter, Local Database creation using Flutter, re-usable code libraries for commonly required functionalities are included with examples. Mobile Application Security is covered briefly along with the link to complete guide available on <https://digital.nic.in> is also given. Testing, Publishing and Maintenance of apps with recent changes in Play/ App Store policies are covered in the end.

Many NIC Officers, including those from Mobile Competency Centres, have contributed in the preparation of this eBook. The names of main contributors with photographs are included under Contributors section and contact address/ emails for MCC HoDs are given so that NIC officers may seek further guidance from them on any topic related to Mobile App Development. Important web reference links to sites providing useful information for app development is covered too.

2.0	App Development Life Cycle
------------	-----------------------------------

In the early days of the modern smartphone applications era, mobile applications went through a similar evolution as first websites. At first, the applications and sites were wholly contained within themselves and acted as little more than static advertisements for the brand, company, product, or service.

However, as connectivity and network capabilities improved, the applications became increasingly connected to sources of data and information that lived outside of the app itself, and the apps became increasingly dynamic as they were able to update their UI and content with data received over the network from queries to data sources.

As a result, the mobile front-end applications increasingly rely on and integrated with back-end services which provide data to be consumed through the mobile front-end. Such data can include, for example, product information for e-commerce apps or flight info for travel and reservation apps. For a mobile game, the data may include new levels or challenges and scores or avatars from other players.

The mobile front-end obtains the data from the back-end via a variety of service calls such as APIs. In some cases, these APIs may be owned and operated by the same entity developing the mobile application. In other cases, the API may be controlled by a third party and access is granted to the mobile application via a commercial arrangement.

2.1	Approaches to App Development
------------	--------------------------------------

There are four major development approaches when building mobile applications

- Native Mobile Applications
- Progressive Web Applications
- Hybrid Mobile Applications
- Cross-Platform Mobile Applications

Each of these approaches for developing mobile applications has its own set of advantages and disadvantages. When choosing the right development approach for their projects, developers consider the desired user experience, the computing resources and native features required by the app, the development budget, time targets, and resources available to maintain the app.

Native Frameworks: These frameworks are available for development of mobile app meant to be used for a specific Mobile OS only. These are developed using programming languages and tools that are provided by the company that develops the platform and the OS on which they will run. To mention a few, Android app can be developed using Java or Kotlin languages using Android Studio or Eclipse

IDE. Similarly, iOS/Apple mobile app can be written in Objective C or Swift using Xcode IDE provided by Apple. There are some advantages and disadvantages in opting for Native framework:

Advantages

- Fastest among the three frameworks
- Single Mobile Platform to focus
- Distribution through Platform Dependent App Stores
- Most Interactive and Intuitive
- Advantage to use Device Essentials
- Smaller in size as compared to Hybrid Apps

Disadvantages

- Needs separate development effort for each Mobile Platform
- Platform Dependent Language and IDE Knowledge
- Expensive, as it requires Platform Specific Skill set, therefore also difficult to maintain

Progressive Web Mobile App: Such kind of app development is more suitable for the web application developers and is focused for mobile devices. Such apps are accessible through web browsers and do not have easy access to device essentials. Therefore, these do not enjoy the advantages available for a mobile application in terms of availability of various components of mobile device like camera, GPS etc. The pros and cons of using this type of framework would be:

Advantages

- First choice for web developers, being developed using HTML/CSS/JS
- Easy to maintain for web developer being similar to web application
- Freedom of Platform/Language maintains continuity
- Cost Effective as compared to Native and Hybrid Apps as no additional technical manpower required if you have a web developer
- Single App for all Mobile Platforms (being accessible on browser only)

Disadvantages

- Browser based access
- Slower than Native & Hybrid Apps
- Cannot use the Advantages of Mobile Device Components
- Limited UI/UX, Less Intuitive
- Cannot be Installed on Device
- Cannot be hosted on App Store, No Icon based Identification

Hybrid Mobile App Development Framework: It is a blend of Web and Native framework to greater extent. The advantage is of using single source code for multiple platforms with a common code up to 70%. Some example of IDEs for this

framework are Ionic, Apache Cordova are some example. The Pros and Cons of using this platform are similar to that of Cross-Platform but with limitation of Using device components/essentials, is not easy sometimes.

Cross-Platform Mobile App Development Framework: This is the most suitable framework if the focus is to provide mobile app for multiple platforms with single development effort. These apps provide native app like experience and 80-90% of the source code is utilized cross-platform which saves resources in terms of manpower and time. React Native, Xamarin, Flutter, PhoneGap, NativeScript are some of the frameworks available in this category. There are some positives and negatives with this kind of framework also:

Advantages

- Expertise of one language/IDE required
- Cheaper than native Apps framework
- Distribution through Platform Dependent App Stores
- Most Interactive and Intuitive
- Advantage to use Device Components

Disadvantages

- App Size is bigger than native apps (3-4 times)
- Little bit slower as compared to Native Apps
- The choice of UI controls is limited to availability on all mobile platforms to be covered

Advantages of the Mobile App Development: As a Mobile App developer, there are certain advantages as given below:

- **Mobile Device Features:** It enables ease of utilizing features like Camera, GPS, Contact List, SMS, Phone Calls, Sensors etc. These device specific features are easy to use for the user as well as easy for the developer to incorporate. The app developer must mention the permissions required in the app. In case some sensitive data is being captured or sensitive permission is required, app store dependent declaration and privacy policy becomes must.
- **Ability to work offline:** The other major advantage with Mobile App is leverage of local app-based database which can be integrated in Mobile App for online and offline features. There can be an app which provides tool for field functionaries to report progress or even Geo Tag assets or locations. The basic data as per user's role can be provided offline in the app using Web API at a connected location and the data collection can be carried out against this offline data in the field by the app users. This offline data can be sent to central server automatically as and when the user gets connectivity.

2.2 Development Life Cycle

The Mobile App Development Life Cycle starts from initiation/strategy and ends with App publishing followed by regular updates. This is shown in the image below. Some of the steps will be repetitive but following these will help in developing a useful mobile App with good user interface. The actual utility to the user in understanding the App navigation will decide the fate of the App, resulting in better utilization, good reviews and high ratings.



- **Strategy:** The first step includes identifying the app users, researching the competition, establishing the app's goals and objectives. It also covers identifying what would be app target like G2G, G2C, G2B or G2E or may be multiple service delivery types and selecting a mobile platform for the app development.
- **Analysis and Planning:** Identify the requirements and prepare product roadmap. Define use cases covering the complete functionality being planned in the App. After listing the functions, decide priority for every functionality to converge into versions and group Functionalities into Milestones. Decide the App name and check for availability of same or similar names on app store to distinguish from existing app names.
- **UI/UX Design:** Create a seamless and effortless user experience. The most important part of any mobile app is that app functionalities should be easy to use and should be intuitive. A mobile app can be designed based on advanced gesture features like Tap, Swipe, Drag, Hold and more. For example, an app can let users move to a next or previous step using the swipe gesture. The UI/UX has a goal to provide excellent User Experience making the app Interactive, Intuitive & User Friendly and keep the user engaged.

- **Information Architecture:** You must decide the data the App will display and capture, the Transactions/Interactions user will carry in the app, the navigations within the app using the gestures or buttons and the workflows. You need to define the role-based access for the Mobile App and plan limited Roles to start with and add functionalities of more Roles in future versions.
- **Wireframes:** Mobile App designer sketches the screens on paper, keeping in mind the Mobile Screens. Wireframes are digital sketches of these screens and help to understand the look and feel of the App being developed.
- **Standard Style Guide:** Developer will follow the Standards set by the organization, like logos, fonts, colour schema and developer information for feedbacks as email in addition to app store feedback reviews. Standard header/footer, if any, also needs to be placed.
- **Mockups:** The second phase of Wireframes is to apply the Standard Styles on Wireframe to convey a final look and feel of your app where colours, images to be used are also part of the design. This provides an exact screen wise look of the app being planned.
- **Prototypes:** Arranging the mock-ups to produce app functionalities. This helps to go through the App as a prototype (*Simulation*).
- **App Development:** Before actual development/programming efforts start, developer will have to define the technical architecture, pick a technology stack and define the development milestones. A typical mobile app project is made up of three integral parts, **back-end/server technology, API(s) and the mobile app front-end.**
- **UAT & Deployment:** Once the Development is complete, user acceptance testing is carried out and modifications are carried out as per testing feedback. Finally, the app is submitted in the respective platform App Store for publishing and is made available to the user for installation.

The follow up actions include, regular checking / updation of API security, updation of mobile App as per Play/App store policies, changing development environments and of course User Ratings/ Reviews. The Developer must keep checking the reviews on the Stores and reply to User feedback which may need some clarifications. Similarly, addressing issues in the App and publishing updated App at regular intervals is important. You may take access for your email ID to submit replies/ clarifications to these reviews from your Centre of Competence.

2.3	Development Frameworks
------------	-------------------------------

Popular development frameworks for mobile applications include Xamarin, Apache Cordova, Ionic App Development, Flutter etc. Most of the earlier users have been using some of these cross-platform development tools and are comfortable in using these. Flutter is comparatively new tool and easy to use for new developers. A brief description of these frameworks is given below for understanding purpose.

The advantages of using these in specific cases is also highlighted along with their pros and cons.

Xamarin

Xamarin is a tool used for cross-platform mobile app development that allows developers to share up to 90 percent of code across major platforms. Xamarin uses a single language, C#, to create apps for all mobile platforms. Xamarin is natively compiled, which makes it a go-to option for building high-performance apps with native look and feel. Additionally, Xamarin can leverage all native and the latest API access to utilize underlying platform capabilities in Xamarin apps such as ARKit on iOS or Android Multi-Window. While the code related to business logic, database access, and network communication can be shared across all platforms. Xamarin allows you to create a platform-specific UI code layer. Thus, Xamarin cross-platform apps look 100% native on any device, providing a better user experience, as compared to generic hybrid apps.

System Requirements:

Windows

- A computer with at least 2GB of RAM and running Windows 7 or higher (*Windows 8-10 is highly recommended*)
- Visual Studio 2012 Professional or higher
- Xamarin for Visual Studio

Mac

- A Mac computer running OS X Yosemite (10.10) or higher
- Xamarin iOS SDK
- Apple's Xcode (7+) IDE and iOS SDK
- Xamarin Studio

Installation:

- Download the Xamarin Installer from <https://www.xamarin.com/download>
- Before running the Xamarin installer, make sure you have installed Android SDK and Java SDK on your computer.
- Run the downloaded installer to begin the installation process
- The Xamarin license agreement screen appears. Click the Next button to accept the agreement.
- The installer will search for any missing components and prompt you to download and install them.
- After the Xamarin installation is complete, click the Close button to exit and get ready to start using Xamarin.

Pros of Using Xamarin for Development:

- Performance Close to Native: Unlike traditional hybrid solutions based on web technologies, a cross-platform app built with Xamarin can still be

classified as native. As the platform develops, Xamarin performance is constantly being improved to fully match the standards of native development, with Microsoft giving thrust on optimization techniques. Visual Studio also offers a complete solution for building, testing, and tracking the app's performance. Visual Studio App Center allows you to run automated UI tests and identify issues before release. However, this service is provided at an additional fee.

- **Native User Experiences:** Xamarin allows you to create flawless experiences using platform-specific UI elements. It's also possible to build cross-platform apps for iOS, Android, or Windows using Xamarin.Forms tool, which converts app UI components into the platform-specific interface elements at runtime. As the use of Xamarin.Forms significantly increases the speed of app development, it is a great option for business-oriented projects. Yet, there might be a slight decline in performance due to the extra abstraction layer. For custom app UI and higher performance, you can still use Xamarin.iOS and Xamarin.Android separately to ensure excellent results.
- **Full Hardware Support:** With Xamarin, your solution gets native-level app functionality. It eliminates all hardware compatibility issues, using plugins and specific APIs, to work with common devices functionality across the platforms. Along with the access to platform-specific APIs, Xamarin supports linking with native libraries. This allows for better customization and native-level functionality with little overhead.

Cons:

- **High Cost for Professional and Enterprise Use:** Although Xamarin is a free open-source platform for individual developers, the framework may cost a pretty penny for enterprise needs. You will spend a lot to purchase a license for Visual Studio.
- **Limited Access to Open Source Libraries:** Native development makes extensive use of open source technologies. With Xamarin, you have to use the elements provided by the platform and some .NET open source resources.
- **Larger App Size:** Depending on their type and complexity, Xamarin apps are typically larger than native ones (the later might be half the size of a Xamarin app). A simple "hello, world" app for Android might take up to 16 MB, much of it being used by the associated libraries, content, Mono runtime, and Base Class Library (BCL) assemblies. So far small apps, it may not be useful.

Apache Cordova

Apache Cordova provides a way to develop mobile applications using standard web technologies - HTML5, CSS3, and JavaScript. Cordova is simply a JavaScript API, which serves as a wrapper for native code and is consistent across devices.

Cordova is an application container with a web view, which covers the entire screen of the device. The web view used by Cordova is the same web view used by the native operating system. On iOS, this is the default Objective-C UIWebView or a custom WKWebView class; on Android, this is android.webkit.WebView. It comes with a set of pre-developed plugins, which provide access to the device's camera, GPS, file system, etc. Cordova applications install just like native applications. This means that building your code for iOS will produce an IPA file and for Android an APK file.

Installation:

- To develop Android apps, first install the Android SDK, and also Java if not already installed on the machine.
- Follow these steps to install Cordova:
 1. **Install Node.js.** Cordova runs on the Node.js platform, which needs to be installed as the first step. Download installer from <https://nodejs.org>
 2. Go ahead and run the downloaded installation file. It is recommended to use the default settings. Node.js needs to be added to the PATH environment variable, which is done by default.
 3. To test the installation, open a command window (make sure you open a new command window to get the updated path settings made by the Node.js installation), and type:
node --version
If the version number is displayed, Node.js is installed and working!
 4. **Install Git.** Git is a version control system, which is used by Cordova behind-the-scenes. Download and install from <http://git-scm.com>. Default settings are recommended.
 5. **Install Cordova.** Cordova is installed using the Node Package Manager (npm). Type the following in the command window to install:
npm install -g cordova
 6. Test the Cordova install by typing:
cordova -version

Pros of Using Cordova:

- Cost-effective, as there is no need to spend resources on different platforms. Instead, you can use a single code base to create an app for multiple platforms.
- Saves time as hybrid apps takes lesser time to develop as compared to developing an app for each native platform separately.
- There are many community add-ons that can be used with Cordova, these have several libraries and frameworks, which are optimized for working with it.

- Since we are using JavaScript when working with Cordova, we don't need to learn **platform specific programming languages**.

Cons:

- Hybrid apps are slower than native ones, so it is not optimal to use Cordova for large apps that require lots of data and functionality.
- Cross browser compatibility can create lots of issues. Most of the time we are building apps for different platforms so the testing and optimizing can be time consuming since we need to cover large number of devices and operating systems.
- Some plugins have compatibility issues with different devices and platforms. There are also some native APIs that are not yet supported by Cordova.

Ionic App Development

Ionic is an open source UI framework for cross-platform, hybrid app development. It is built on top of standardized web technologies: HTML, CSS and Javascript and enables the skills of web developers to be used to build applications. Ionic's universal web components pair with any JavaScript framework, including Angular, React, Vue, or no framework at all. It is also back-end agnostic, with connections to AWS, Azure, and Firebase.

Installation:

- Before proceeding, make sure the machine has Node.js and git installed.
- Install the Ionic CLI with npm: `npm install -g @ionic/cli`
- Start an app: `ionic start`
- Run an app: `ionic serve`

Pros of Using Ionic Apps Development:

- Can develop for both iOS and Android at once (with some restrictions such as platform support for styling and plugins).
- Plenty of UI component available and easy to use — cards, buttons, toggles, segments, modals, inputs, lists, row/column grid, etc.
- Lots of plugins that allow the use of smartphone - features like cameras, fingerprint scanners, NFC, geolocation, sending analytics to Firebase, push notifications, and deep links.
- Angular, HTML, CSS, and JavaScript skills are (almost) all you need to get started — no need for Java and Swift or Objective-C

Cons:

- Native plugins aren't stable and can conflict with each other, but necessary features are available in Ionic by default.
- Debugging of an application built on Ionic can be challenging and requires more time because of unclear error messages.

- Builds can randomly **crash** without any reason

The existing mobile app developers may be using one or the other framework and will be comfortable in their development work. But new users may use Flutter as the recommended framework due to certain advantages which are listed in the next section.

2.4	Flutter - Recommended Framework
------------	--

Flutter is free and open-source Google's mobile UI framework for crafting high-quality native interfaces on iOS, Android, Web and desktop. Flutter makes an ideal choice for those venturing into fresh app development, making it grow as a compatible cross-platform framework leaving behind all its alternatives. Following reasons justify using Flutter as first choice for new NIC Developers:

- **Open Source:** Flutter is an open-source code software development toolkit from Google. It provides easy posting of issues and access to documentation from open developer forums.
- **Single Codebase:** Being a cross-platform framework, it allows programmers to write code once and they can use it on multiple platforms. This means that a single version of an application runs on both iOS and Android. This saves a lot of time and effort in writing code for different platforms, as with native frameworks.
- **Dart as Programming Language:** Flutter uses Dart as an object-oriented programming language to create apps. The prominent features of Dart include a rich standard library, garbage collection, strong typing, generics, and async-awaits.
- **Performance:** Flutter application is built directly into the machine code, which eliminates any performance bugs of the interpretation process. It improves the performance of the application as compared to any other app development platform.
- **Hot Reload and Development:** Developers can see the changes made to code instantly. Any updates are available to both the designers and developers in a matter of seconds.
- **Custom Widgets:** Flutter offers a myriad of widgets to help developers in their creation process. With Flutter, you can simply fabricate existing widgets along with customization that empowers you to create responsive and fascinating portable applications.
- **Requires Less Testing:** Normally testing would require checking on compatibility on different platforms. With Flutter, apps use a single code base with no change to run across different platforms. All one needs to do is to test a Flutter application just once and save a lot of time and money for the developer.
- **Potential Ability to Go Beyond Mobile** There's also Flutter for Web and Flutter Desktop Embeddings now. Flutter Web that makes it possible to run

pure Flutter applications in a browser without modifying the source-code. This marks Flutter's transition from a cross-platform mobile application framework to a full-blown cross-platform development tool.

- **IoT Applications with Flutter:** Flutter really stands apart from all the other frameworks as Flutter SDK can easily be integrated with the Internet of Things (IoT) (as it can enable you to create the modern app).
- **It has support for** storage, camera, location, network, and more.

2.5	Targeted Users and Purpose
------------	-----------------------------------

All mobile apps should include the targeted user and purpose of the app on first screen. Although the details are always mentioned on the play store listing but as a best practice we must include this detail. This is specifically required in apps where no content is accessible without authentication and presently the first screen asks for Login Password without any introductory content for the user.

In case the app is meant for the authorised user only, we can include the Term and Conditions screen as the first screen mentioning the details of the app and targeted user along with the declaration by the user, admitting that user is authorized to use the app.

This will help the unintended common play store user who may have installed the app on personal device but may not be meant for that user. This will also help in avoiding negative user reviews.

2.6	Wireframe of Mobile App
------------	--------------------------------

Mobile User Interface (UI) design is quite essential. Mobile UI considers constraints and contexts, screen, input and mobility as outlines for design. The user is often the focus of interaction with their device, and the interface entails both hardware and software components. User input allows for the users to manipulate a system, and device's output allows the system to indicate the effects of the users' manipulation. Mobile UI design constraints include limited attention and form factors, such as a mobile device's screen size for a user's hand. Mobile UI contexts signal cues from user activity, such as location and scheduling that can be shown from user interactions within a mobile application. Overall, mobile UI design's goal is primarily for an understandable, user-friendly interface.

It is a good idea to design a wireframe of the mobile app to be developed. Wireframing is the early step of the **UI/UX design** process when the structure of the App is being framed. The usability and efficiency of the final app often depend on how well the wireframe is created at the very first steps of the design process. It is basically a schematic or blueprint that is useful for helping in communicating/showcasing or demonstrating the likely functionality of the mobile app to be developed. Many tools are available on the Internet but the Adobe XD (also known

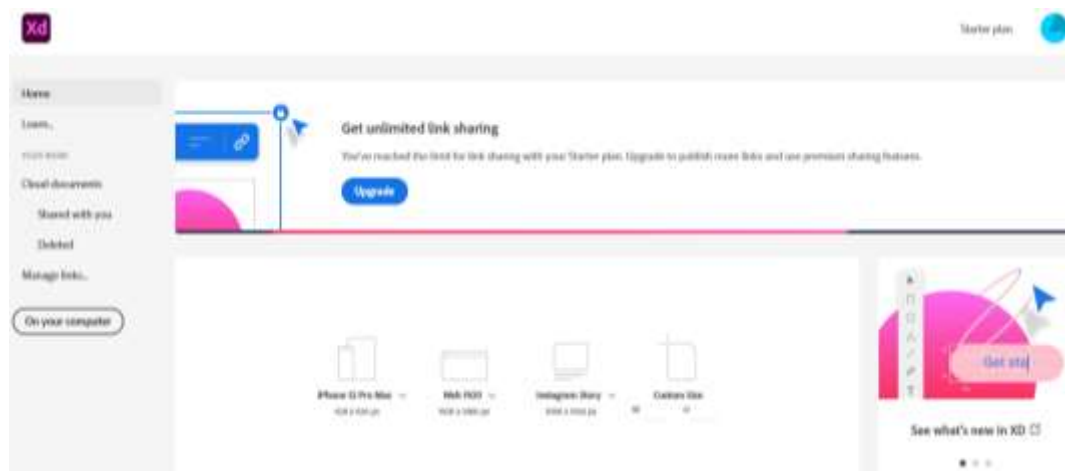
as Adobe Experience Design) can be used. It is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. Adobe XD is the Adobe prototyping tool for user experience and interaction designers. Its features are used for creating wireframes, prototypes, and screen designs for digital products such as websites and mobile apps.

You can install it, as per steps given below:

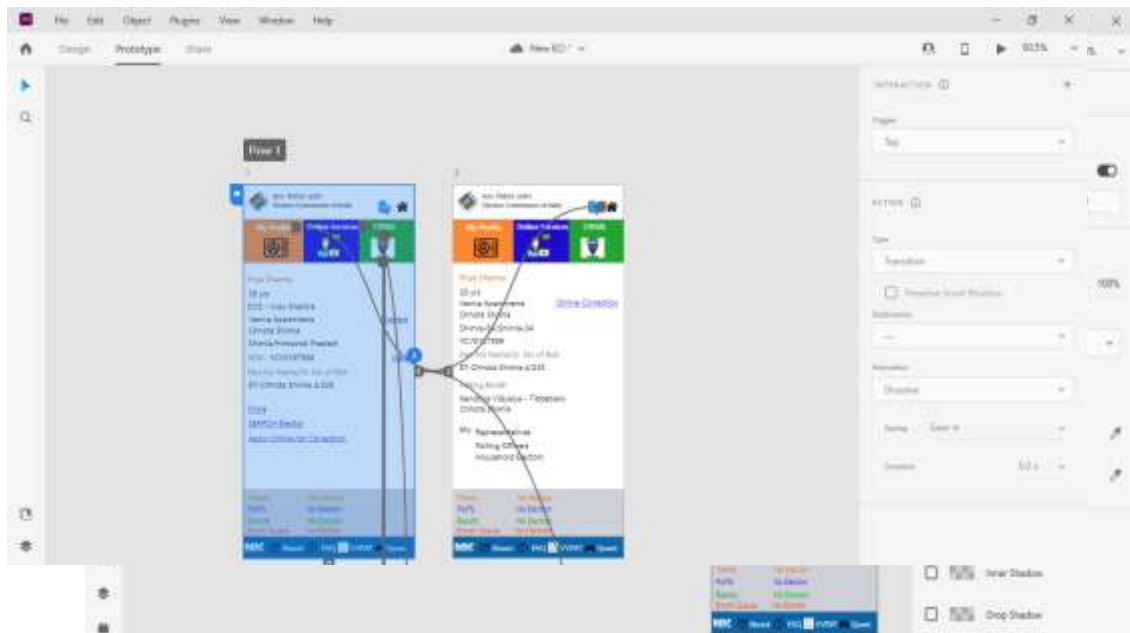
1. Download and install ADOBE CREATIVE CLOUD.
2. After installing creative cloud, download and install ADOBE XD.
3. After installing XD choose the screen size for wireframe of mobile or web.
4. There are two tabs in XD, Design and Prototype. Design part is done through Design tab and screens are connected with each other through prototype.
5. In Design part user can add the design from UI kits which are preinstalled for windows and mobile.
6. Icons and shapes can be designed or downloaded from Google and can be imported to project.
7. Once all the work is done, user can export the art board to different formats such as png, jpeg etc.

Adobe XD Work Flow:

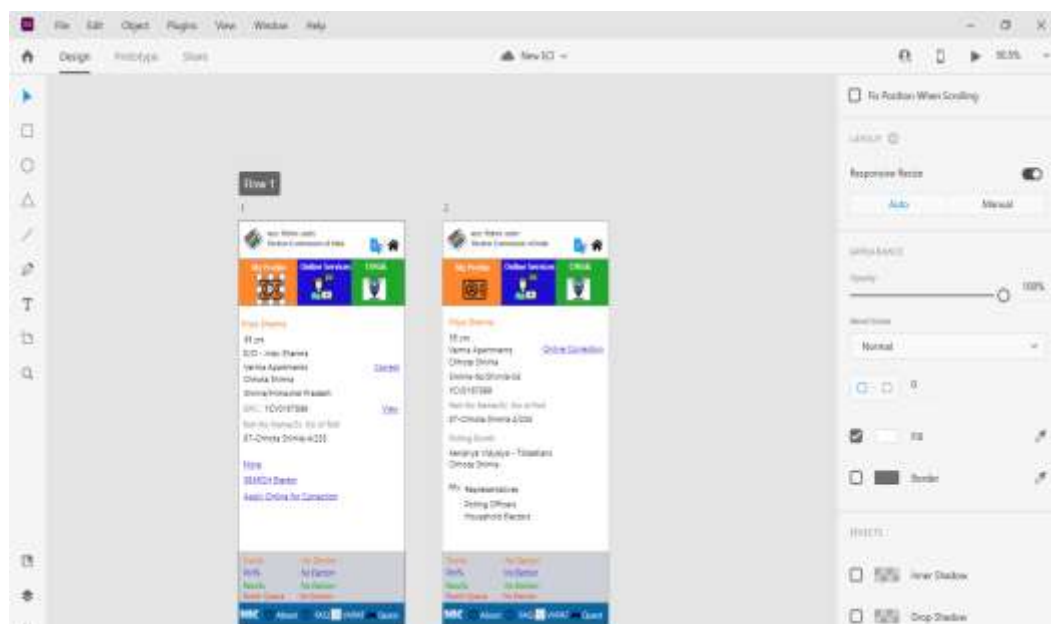
1. Open ADOBE XD and a screen will appear where user has to choose the screen size or frame for the wireframe to be developed. (We have chosen iPhone 12 pro max 428*926 px).



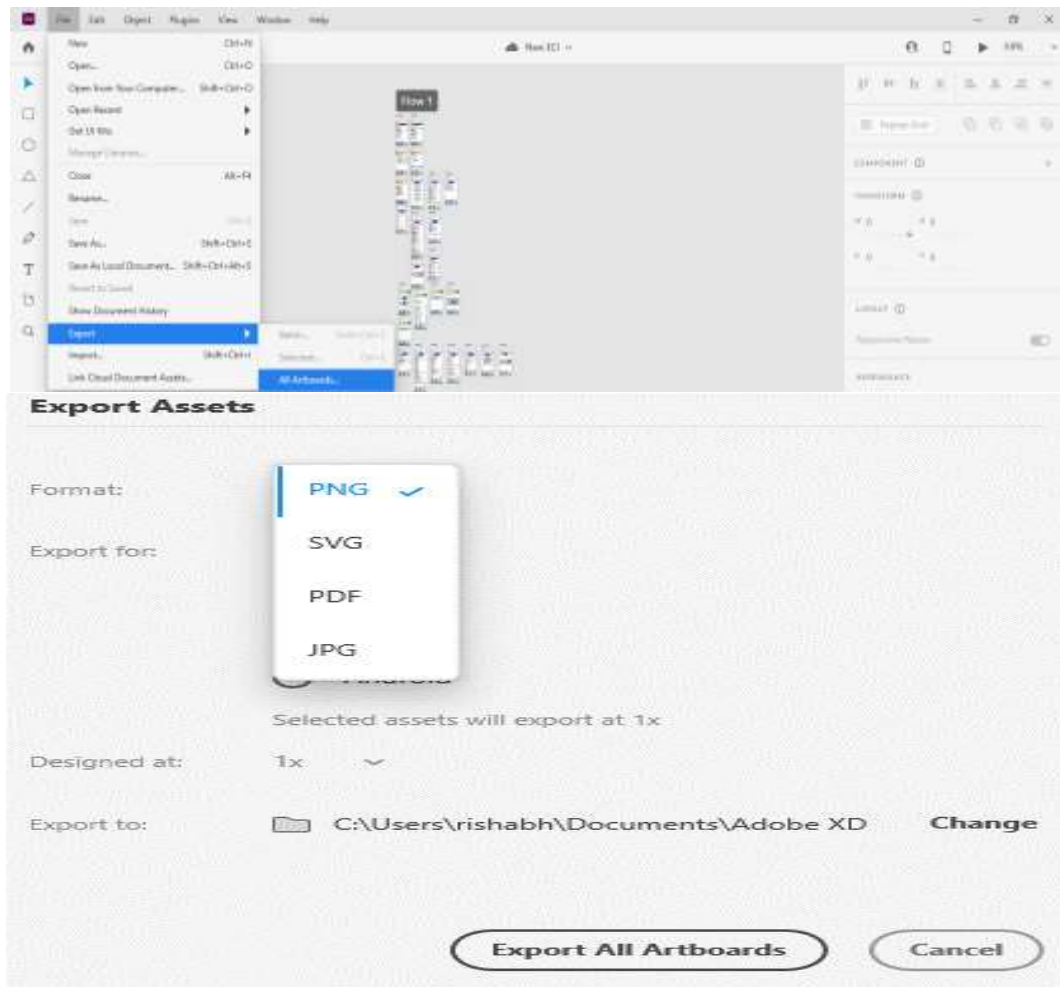
2. Next step is the **design** part. In the screen shown below, showing the App image, it is developed by using the tools which are available in the left panel of the screen such as pointer, text, line etc. Colours are available in the right panel in the screen. Icons are downloaded through Google.



3. Next step is to create next screen of the application by clicking on the **numeric1** on the top of the screen number 1 and press Ctrl button from the keyboard and drag the screen corresponding to the first screen, which creates a copy of the existing screen. Make the required changes on new screen as per the requirement. Similarly, any number of screens can be created as per requirement.
4. To make clickable events for the screen, it is done in **prototype**. Click on the button, screen, text or dropdown menus from the 1st screen and drag the blue dot to the 2nd screen. This is how proto typing is done.



5. After all screens are developed, user can export the screens in to different formats png, svg, pdf, jpg and click on export all artboard button. Using screen capture tool, a video can be made showing the flow of mobile app when different buttons are pressed.



3.0	Flutter as Recommended Framework
------------	---

Flutter applications are performant by default, but to gain the maximum performance from application, some good practices to follow while developing app:

1. Don't make costly operations such as async calls (network request) or too big calculations inside build method, this can lead to performance loss.
2. Split the code into small widgets which increase the reusability of widgets as well as less code to maintain.
3. Use absolute positioning only when it's truly necessary as flutter applications run on various devices and not all devices have the same pixel density, size, etc.
4. Use "Const" keyword whenever possible while building widgets which helps Flutter to rebuild only those widgets that should be updated.
5. Most importantly use only necessary packages/libraries, this will not only improve app performance but also significantly reduces the size of the application.

Essential Packages/Libraries

- **http:** This package assists in the network related tasks such as get, post, update, delete request with minimal hassle.
 - o URL: <https://pub.dev/packages/http>
- **sqlite:** Plugin for SQLite helps to maintain local database for the app.
 - o URL: <https://pub.dev/packages/sqlite>
- **url_launcher:** This package is used to launch URLs inside mobile WebView. It supports multiple URL schemes like HTTP, mailto, SMS, phone dialler etc.
 - o URL: https://pub.dev/packages/url_launcher
- **path_provider:** This package helps developers to get location on Android and iOS file systems such as temp or app data directories. It supports both internal and external storage and makes it easy to get directories.
 - o URL: https://pub.dev/packages/path_provider
- **local_auth:** This package helps to implement local and biometric authentication on the user's device, such as fingerprint authentication and face biometric authentication.
 - o URL: https://pub.dev/packages/local_auth
- **cached_network_image:** This package is used to show images from the internet and keep them in cache directory for offline showing.
 - o URL: https://pub.dev/packages/cached_network_image

- **image_picker:** This package is used for selecting images from gallery or camera on both iOS and android.
 - o URL: https://pub.dev/packages/image_picker
- **fl_chart:** This package is used to draw charts on screen such as pie charts, bar charts, line charts, etc. It also supports customization of looks and feel of the graphs to develop data-intensive apps.
 - o URL: https://pub.dev/packages/fl_chart
- **shared_preferences:** This package provides platform-specific persistent storage libraries to store small data such as user info, login details, etc.
 - o URL: https://pub.dev/packages/shared_preferences
- **location:** This package makes it easy to get access to the current location of the device. It also provides call backs when the location is changed.
 - o URL: <https://pub.dev/packages/location>
- **flutter_launcher_icons:** This package simplifies the task of updating app's launcher icon, include it under dev_dependencies property of pubspec.yaml file.
 URL: https://pub.dev/packages/flutter_launcher_icons

Security Aspect

- Stay up-to-date: Keep the Flutter SDK, plugins, and packages up-to-date is the easiest and one of the best ways to secure the app.
- Obfuscate code: It is the practice of making code difficult to understand using reverse engineering.
- Use -obfuscate argument while building the android app bundle or apk.
- Limit permissions: Ask for that permissions only which the app required to work efficiently. Also, in choosing a package, one should always check whether the package has dubious permission request or not.
- Secure user data: Sometimes personal info about user has to be stored inside the app for offline access. In that case, use flutter_secure_storage package for storing sensitive data.
- Integrate local authentication: Use local_auth package to enhance the security of the app through biometric authentication.
- Jailbroken and rooted devices support: These devices have more privileges so it is always a good practice to detect if the app is running on a jailbroken or rooted device. Flutter jailbreak detection package provides support to detect these kinds of devices.

3.1	Setting up Flutter Environment
------------	---------------------------------------

Use the following link to install Flutter SDK as per the OS:

URL: <https://flutter.dev/docs/get-started/install>

Setup:

Step 1: Open the terminal and run “flutter create” command:

```
$ flutter create myapp
$ cd myapp
```

This creates a new project with myapp name which contains all material components.

Step 2: Use the “flutter run” command to run this app in emulator.

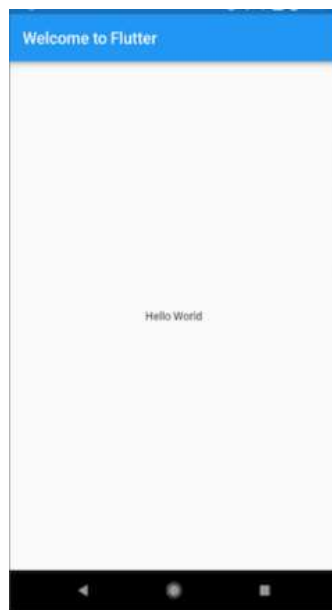
Step 3: Edit lib/main.dart file which is a starting point of the app.

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Welcome to Flutter'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Step 4: By running this code, following screen is shown in emulator:



Step 5: How to use an external package:

The “pubspec.yaml” file manages all the assets and dependencies of the app. Under the dependencies section, add package name along with version number, if version number is not known leave it blank.

```
dependencies:  
  flutter:  
    sdk: flutter  
  sqflite: ^1.3.0  
  progress_dialog: ^1.2.4  
  
  path_provider: ^1.6.24  
  http: ^0.12.1  
  wc_form_validators: ^0.1.1  
  intl:
```

Step 6: Run “flutter pub get” command: It automatically fetches the required files from internet.

Step 7: Import new package:

Inside lib/main.dart file, write the following line:

```
import 'package:path_provider/path_provider.dart';
```

3.2	Demo App Tutorial using SQLITE and API
------------	---

1. Create new Flutter project as mentioned in above steps.

2. In this app we are using the open source API, the link is:
<https://jsonplaceholder.typicode.com/users>

```
{
  id: 1,
  name: "Leanne Graham",
  username: "Bret",
  email: "Sincere@april.biz",
  + address: {...},
  phone: "1-770-736-8031 x56442",
  website: "hildegard.org",
  + company: {...}
},
```

3. Create the user model class: To represent information provided by api, we use User class.

```
List<User> userFromJson(String str) =>
    List<User>.from(json.decode(str).map((x) => User.fromJson(x)));

String userToJson(List<User> data) =>
    json.encode(List<dynamic>.from(data.map((x) => x.toJson())));

class User {
  User({
    this.id,
    this.name,
    this.username,
    this.email,
    this.phone,
    this.website,
  });

  int id;
  String name;
  String username;
  String email;
  String phone;
  String website;

  factory User.fromJson(Map<String, dynamic> json) => User(
    id: json["id"],
    name: json["name"],
    username: json["username"],
    email: json["email"],
    phone: json["phone"],
    website: json["website"],
  );

  Map<String, dynamic> toJson() => {
    "id": id,
    "name": name,
    "username": username,
    "email": email,
    "phone": phone,
    "website": website,
  };
}
```

4. Create sqlite database:

Add "sqlite", "path_provider" and "http" package in pubspec.yaml file.

```
dependencies:  
  flutter:  
    sdk: flutter  
  sqflite:  
  path_provider:  
  http:
```

Define database related functions in "database_helper.dart" file.

```
class DatabaseHelper {  
  static DatabaseHelper _databaseHelper; // Singleton DatabaseHelper  
  static Database _database;  
  
  DatabaseHelper._createInstance(); // Named constructor to create instance of  
  DatabaseHelper  
  
  factory DatabaseHelper() {  
    if (_databaseHelper == null) {  
      _databaseHelper = DatabaseHelper  
      | | . _createInstance(); // This is executed only once, singleton object  
    }  
    return _databaseHelper;  
  }  
  
  Future<Database> get database async {  
    if (_database == null) {  
      | _database = await initializeDatabase();  
    }  
    return _database;  
  }  
  
  Future<Database> initializeDatabase() async {  
    // Get the directory path for both Android and iOS to store database.  
    Directory directory = await getApplicationDocumentsDirectory();  
    String path = directory.path + 'postDetails.db';  
  
    // Open/create the database at a given path  
    var postDetailsDatabase =  
    | | await openDatabase(path, version: 1, onCreate: _createDb);  
    return postDetailsDatabase;  
  }  
  
  void _createDb(Database db, int newVersion) async {  
    await db.execute("CREATE TABLE IF NOT EXISTS user("  
    | "colId INTEGER PRIMARY KEY AUTOINCREMENT,"  
    | "name TEXT ,"  
    | "id INTEGER,"  
    | "username TEXT ,"  
    | "email TEXT,"  
    | "phone TEXT,"  
    | "website TEXT"  
    | ")");  
  }
```

Insert data into User table:

```
Future<int> insertIntoUserTable(User obj) async {
    Database db = await this.database;

    var result = await db.rawQuery(
        'INSERT INTO user(name, id, username, email, phone, website) VALUES(?, ?, ?, ?, ?, ?)',
        [obj.name, obj.id, obj.username, obj.email, obj.phone, obj.website]);
    // print('data inserted');
    return result;
}
```

Update user table:

```
Future<int> updateUser(User obj) async {
    var db = await this.database;
    var result = await db
        .update('user', obj.toJson(), where: 'colId = ?', whereArgs: [obj.id]);
    return result;
}
```

Delete user from table:

```
Future<int> deleteUser(int id) async {
    var db = await this.database;
    int result = await db.rawQuery('DELETE FROM user WHERE colId = $id');
    return result;
}
```

Get data from User table:

```
Future<List<User>> getDataFromUserTable() async {
    var db = await this.database;
    var results = await db.rawQuery('SELECT * FROM user');
    int count = results.length;

    List<User> userList = [];
    for (int i = 0; i < count; i++) {
        userList.add(User.fromJson(results[i]));
    }
    print(userList.length);
    return userList;
}
```

5. Method to get API data and save to the local database:

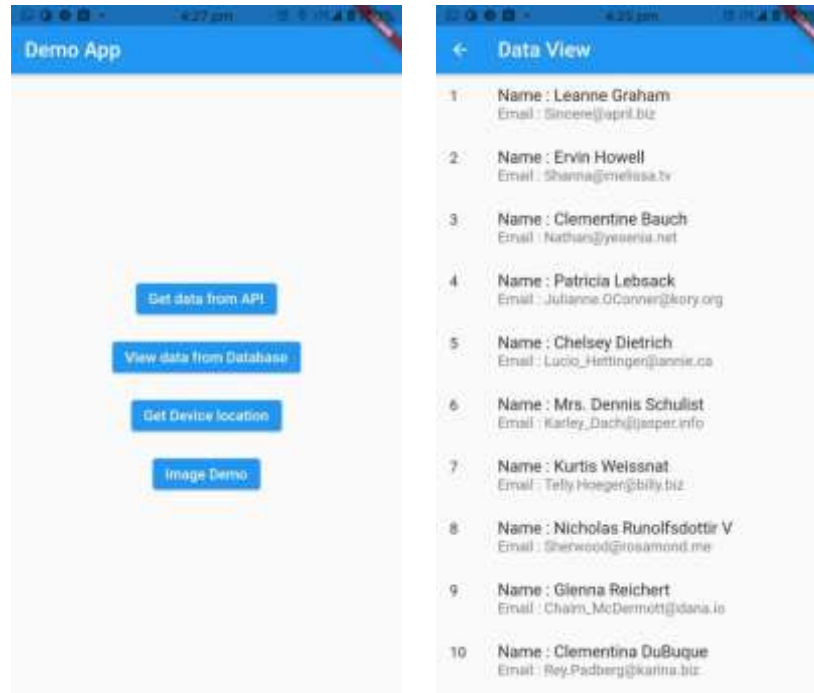
```
Future<List<User>> getUsersFromApi() async {
  var url = "https://jsonplaceholder.typicode.com/users";
  List<User> users = [];
  try {
    var response = await http.get(Uri.parse(url));
    if (response.statusCode == 200 && response.body != null) {
      users = userFromJson(response.body);
      return users;
    }
  } catch (e) {
    print('Exception $e');
  }
}

saveUsersToDatabase(List<User> users) async {
  if (users != null) {
    for (var user in users) {
      await databaseHelper.insertIntoUserTable(user);
    }
  }
}
```

6. Display saved data on a page: Here we use "FutureBuilder" to get data from database and display on a screen.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Data View'),
    ), // AppBar
    body: Container(
      child: FutureBuilder(
        future: data,
        builder: (context, snapshot) {
          switch (snapshot.connectionState) {
            case ConnectionState.none:
              return Center(
                child: CircularProgressIndicator(),
              ); // Center
            case ConnectionState.waiting:
              return Center(
                child: CircularProgressIndicator(),
              ); // Center
            case ConnectionState.done:
              if (snapshot.hasData) {
                return Container(
                  child: ListView.builder(
                    itemCount: snapshot.data.length,
                    itemBuilder: (context, index) {
                      return ListTile(
                        leading:
                          Text('${snapshot.data[index].id.toString()}'),
                        title: Text('Name : ${snapshot.data[index].name}'),
                        subtitle:
                          Text('Email : ${snapshot.data[index].email}'),
                      ); // ListTile
                    }, // ListView.builder
                  ); // Container
              } else {
                return Container(
                  child: Center(
                    child: Text('no data'),
                  ), // Center
                ); // Container
              }
            }
          )
        ),
      ),
    );
}
```

Screen shots of app:



Use Camera and Gallery to pick image

1. Add "image_picker" library in pubspec.yaml file under dependencies section and import this package.
2. Use ImagePicker() method from above library for image selection. To click image from camera, use source as ImageSource.camera, To select image from gallery, use source as ImageSource.gallery, To restrict the size of image, use maxHeight, maxWidth, and imageQuality property.

```
File file;  
Future getImageFromCamera() async {  
  final pickedFile = await ImagePicker().pickImage(  
    source: ImageSource.camera,  
    maxHeight: 600,  
    maxWidth: 800,  
    imageQuality: 80);  
  
  setState(() {  
    file = File(pickedFile.path);  
    int sizeInBytes = file.lengthSync();  
    fileSizeInKbCamera = (sizeInBytes / (1024)).ceil();  
  });  
}
```

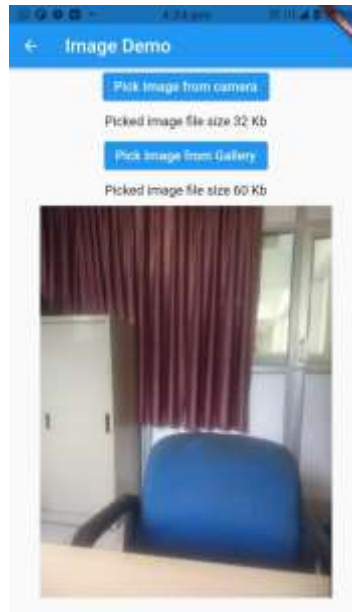
```

Future getImageFromGallery() async {
  final pickedFile = await ImagePicker().pickImage(
    source: ImageSource.gallery,
    maxHeight: 500,
    maxWidth: 600,
    imageQuality: 80);

  setState(() {
    file = File(pickedFile.path);
    int sizeInBytes = file.lengthSync();
    fileSizeInKbGallery = (sizeInBytes / (1024)).ceil();
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('User Location'),
    ), // AppBar
    body: SingleChildScrollView(
      child: Center(
        child: Column(
          children: [
            ElevatedButton(
              onPressed: getImageFromCamera,
              child: Text('Pick Image from camera')), // ElevatedButton
            SizedBox(
              height: 10,
            ), // SizedBox
            Text('Picked image file size $fileSizeInKbCamera Kb'),
            SizedBox(
              height: 10,
            ), // SizedBox
            ElevatedButton(
              onPressed: getImageFromGallery,
              child: Text('Pick Image from Gallery')), // ElevatedButton
            SizedBox(
              height: 10,
            ), // SizedBox
            Text('Picked image file size $fileSizeInKbGallery Kb'),
            SizedBox(
              height: 10,
            ), // SizedBox
            if (file != null)
              Container(
                height: 500,
                width: 400,
                child: Image.file(file),
              ), // Container
          ],
        ), // Column
      ), // Center
    ), // SingleChildScrollView
  ); // Scaffold
}

```

Use GPS

1. Add "location" package inside pubspec.yaml file and import this package.

Android:

To use location background mode on Android, you have to use the `enableBackgroundMode({bool enable})` API before accessing location in the background and adding necessary permissions. You should place the required permissions in your applications

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
```

IOS:

And to use it in iOS, you have to add this permission in Info.plist :

```
// This is probably the only one you need. Background location is supported
// by this -- the caveat is that a blue badge is shown in the status bar
// when the app is using location service while in the background.
NSLocationWhenInUseUsageDescription

// Deprecated, use NSLocationAlwaysAndWhenInUseUsageDescription instead.
NSLocationAlwaysUsageDescription

// Use this very carefully. This key is required only if your iOS app
// uses APIs that access the user's location information at all times,
// even if the app isn't running.
NSLocationAlwaysAndWhenInUseUsageDescription
```

2. Define the getLoc() method to ask for location access, if not granted and get current position of device.

```
getLoc() async {  
  bool _serviceEnabled;  
  PermissionStatus _permissionGranted;  
  
  _serviceEnabled = await location.serviceEnabled();  
  if (!_serviceEnabled) {  
    _serviceEnabled = await location.requestService();  
    if (!_serviceEnabled) {  
      return;  
    }  
  }  
  
  _permissionGranted = await location.hasPermission();  
  if (_permissionGranted == PermissionStatus.denied) {  
    _permissionGranted = await location.requestPermission();  
    if (_permissionGranted != PermissionStatus.granted) {  
      return;  
    }  
  }  
  
  _currentPosition = await location.getLocation();  
  
  location.onLocationChanged.listen((LocationData currentLocation) {  
    print("${currentLocation.longitude} : ${currentLocation.latitude}");  
    setState(() {  
      _currentPosition = currentLocation;  
    });  
  });  
}
```

3. To display latitude and longitude on screen

```
override  
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(  
      title: Text('User location'),  
    ), // AppBar  
    body: Container(  
      child: Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: [  
            if (_currentPosition != null)  
              Text(  
                "Latitude: ${_currentPosition.latitude},\n Longitude: ${  
                  _currentPosition.longitude}  
                }, style: TextStyle(  
                  fontSize: 22,  
                  color: Colors.black,  
                  fontWeight: FontWeight.bold), // TextStyle  
              ), // Text  
          ],  
        ), // Column  
      ), // Center  
    ), // Container  
  ); // Scaffold  
}
```



Competency Centre Tamil Nadu is working on the common usable 10 components for mobile app development using flutter. The list of components being developed are:

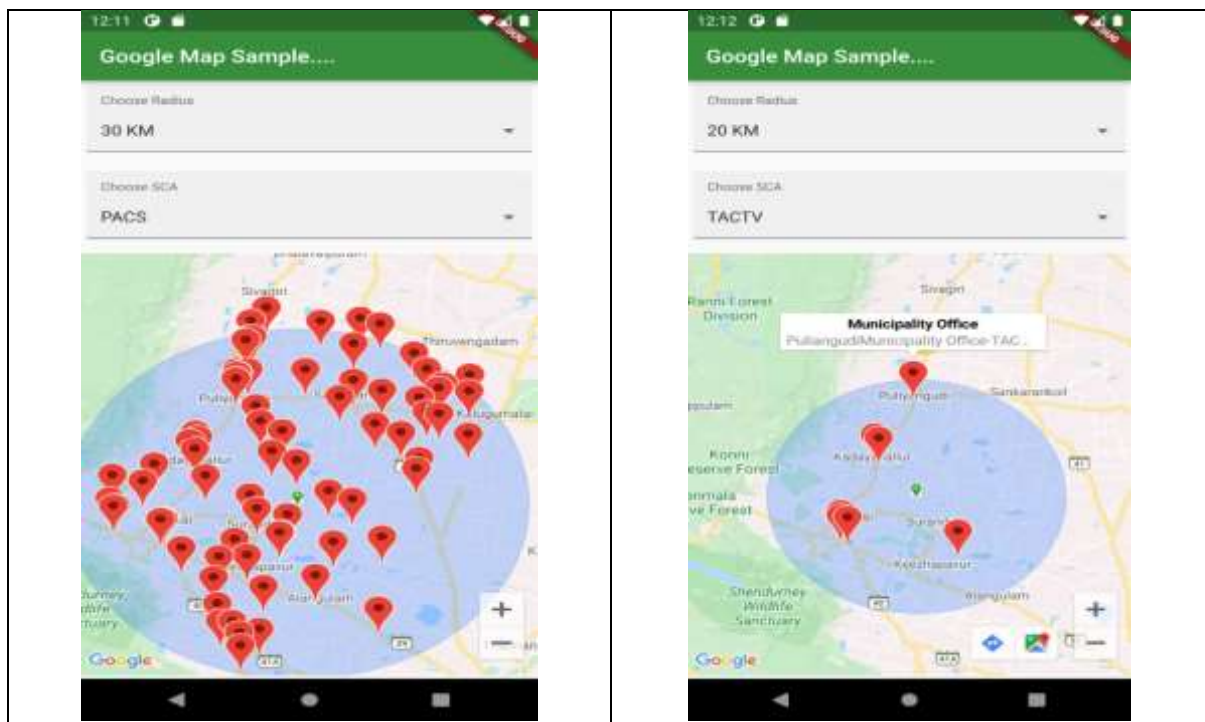
1. Open Street map Integration- display multiple geo coordinates in Open Street map
2. Google map Integration- display multiple geo coordinates in Google map
3. Multiple level menus
4. Photo Gallery from local image file - with zoom option
5. Flutter Charts
6. Data Table
7. Device info – Hardware status/connectivity status
8. Login Screen
9. Photo capture with Geo tagging
10. Form Templates

3.3	Re-Usable Code Libraries
------------	---------------------------------

The reusable component source code can be obtained from the Mobile Competency Centre Tamil Nadu by dropping an email at james.arulraj@nic.in. The general content of already developed 8 components is as follows:

1. Google map Integration- display multiple geo coordinates in Google map

- ❖ Option to view nearby Services centres/Govt establishments within specified range of distance in Google map
- ❖ Different colour marker for Service centres and the current location.
- ❖ Map info window on markers.
- ❖ Input data for Geo coordinates and other location details are in JSON format
- ❖ Zero Coding.
- ❖ The App Name, Labels, Header, info window and other details can be configured in the config.json file.



2. Open Street map Integration

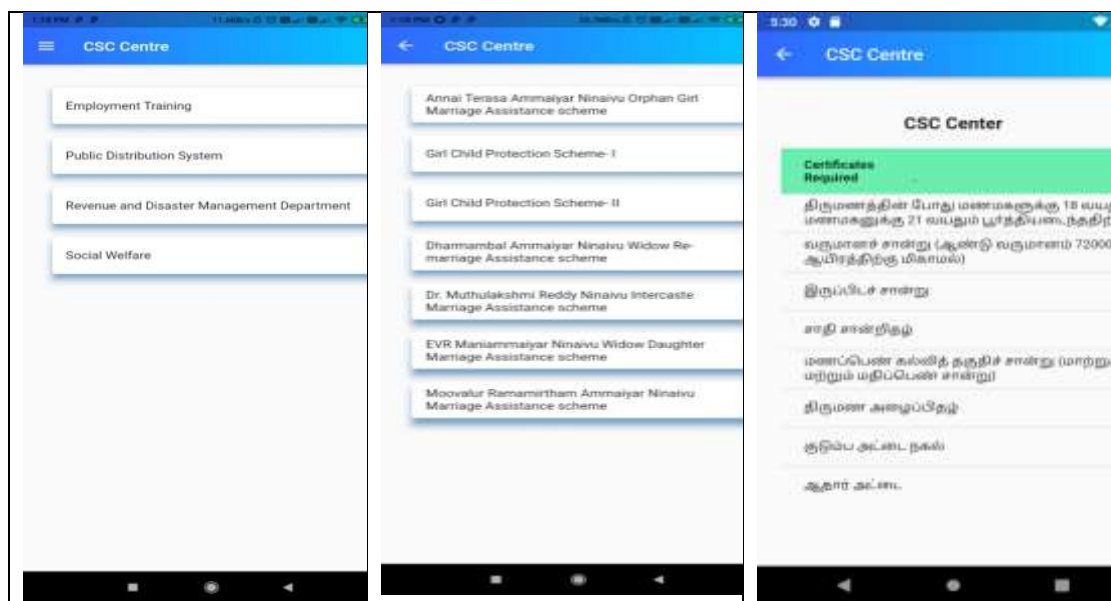
Display multiple geo coordinates on Open Street map

- ❖ Option to view nearby Services centres/Govt establishments within specified range of distance in Google map
- ❖ Different colour marker for Service centres and the current location.
- ❖ Map info window on markers.
- ❖ Input data for Geo coordinates and other location details are in JSON format
- ❖ Zero Coding.
- ❖ The App Name, Labels, Header, info window, data source and other details can be configured in the config file.



3. Multiple Level menus

- ❖ Mobile apps like Citizen Charter/ Schemes details can be developed without coding.
- ❖ Input data is in JSON format
- ❖ Multi level menus
- ❖ Zero Coding
- ❖ The App Name, Labels, Header, Menu names, and data source can be configured in the config file.



4. Data Table

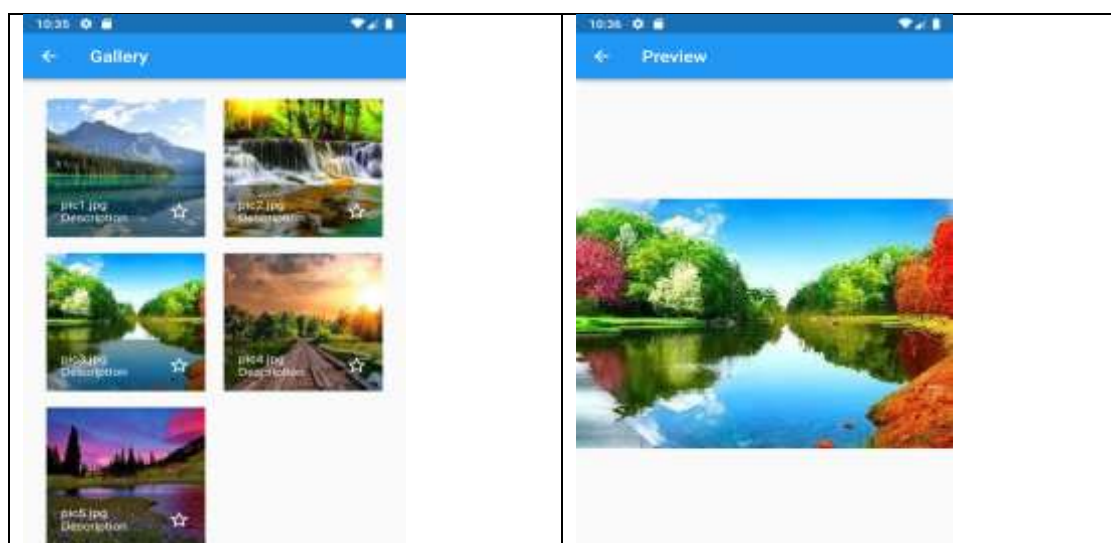
- ❖ Displays the data in Table format
- ❖ Input data is in JSON format
- ❖ The columns to be displayed, Title, report header, column header data source can be configured in the config file without coding
- ❖ Option to display only selected columns

Sample DataTable		
Monthly Report		
Name	Perultrale month	Previous month
Registration	28887	28380
Hypertension	15470	13140
Diabetes Mellitus	5996	5726
Cervical Cancer	1398	1528
Breast Cancer	2129	2210
Other Blood Tests	575	496
ECHO	110	48
ECG	372	381
Gynecology Test	0	0
Pathology Test	2	2
Radiology Test	0	0
Treatment - HT & DM	485	594
Treatment and CaCa	0	0

Sample DataTable			
Monthly Report			
Perultrale month	Previous month	Current month	Total count
28380	13058	5863289	
13140	5215	3855087	
5726	2226	1493059	
1528	578	896598	
2210	825	779228	
496	155	198842	
48	8	36309	
381	96	71250	
0	2	1670	
2	2	1972	
0	0	587	
594	33	67247	
0	0	210	

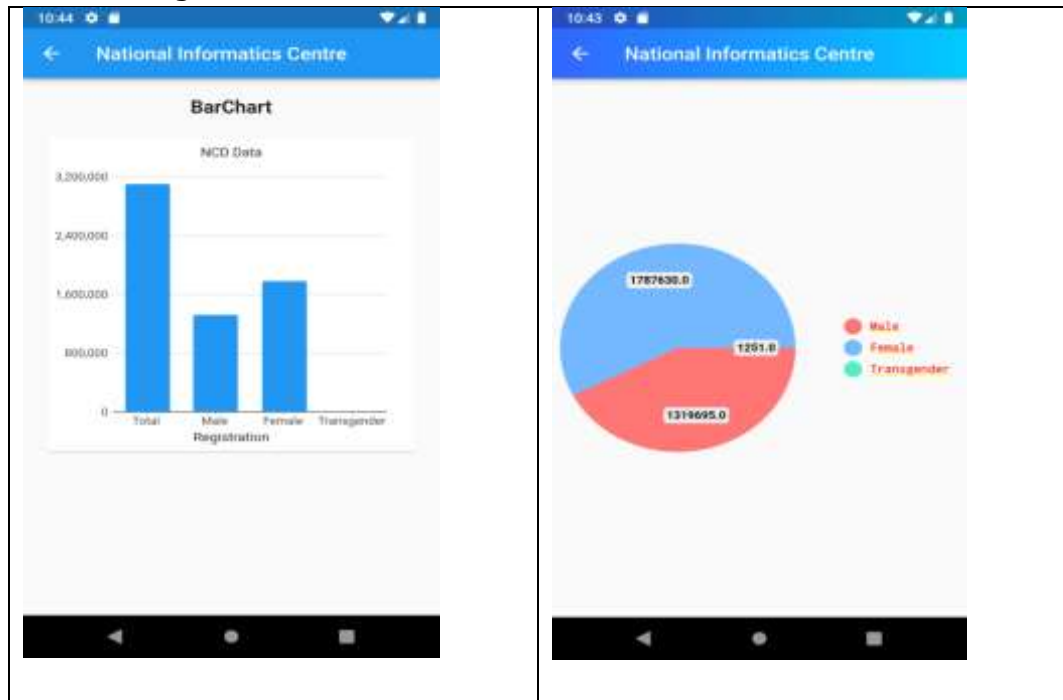
5. Photo Gallery from local image file - with zoom option

- ❖ Displays photo in grid format
- ❖ Option to zoom/enlarge the thumbnail.
- ❖ Simply copy the Photos in Assets folder and configure the config.json file
- ❖ No coding required.



6. Flutter Charts

- ❖ Pie Chart and Bar charts are available
- ❖ Data source in Json format
- ❖ No coding required.
- ❖ The Chart type, header and data source can be configured in the config file



7. Device info – Hardware status

Following detail about the mobile device information can be viewed

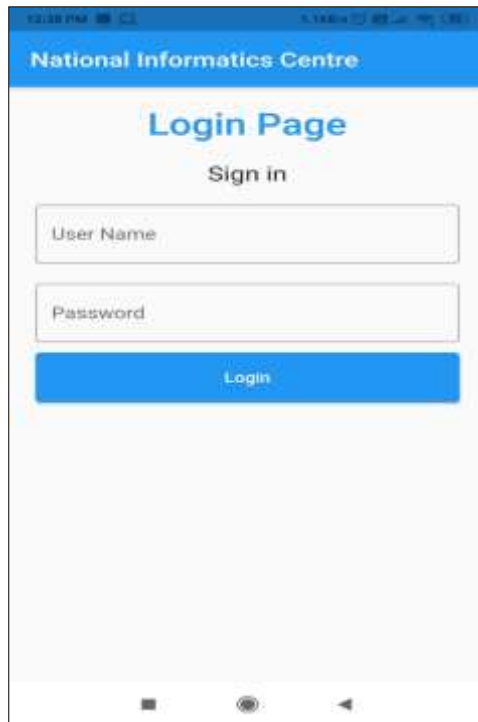
- ❖ Device Make and Model
- ❖ RAM Size
- ❖ Internal memory available
- ❖ Device id/ IMEI No.
- ❖ No coding required

The screenshot shows the 'Device Info' screen of the National Informatics Centre app. The screen displays a list of device information:

Device Info	
App Name	device_info
App Version	1.0.0
Device Id	8B19D0662C7006d
Build No	9903120112002
IMEI No	
Android Version	11
Device Model	LE2104
Device Manufacturer	OnePlus
Internal Memory Free Size	92.28 GB
External Memory Free Size	0.0 GB
RAM Size	7521 GB
Device Current Date and Time	2021-11-08 03:40

8. Login Screen - User Name / Password

- ❖ Login screen
- ❖ The header, label and button text can be configured in the config file
- ❖ No coding required

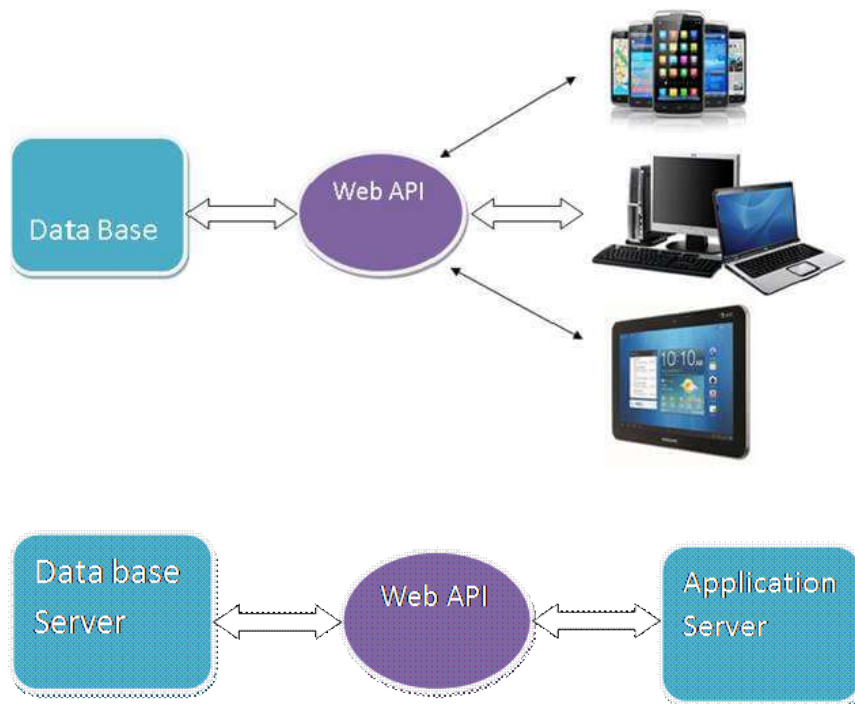


4.0 Web APIs for Communication

Web API

An Application Programming Interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.

To put it in simple terms, API is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.

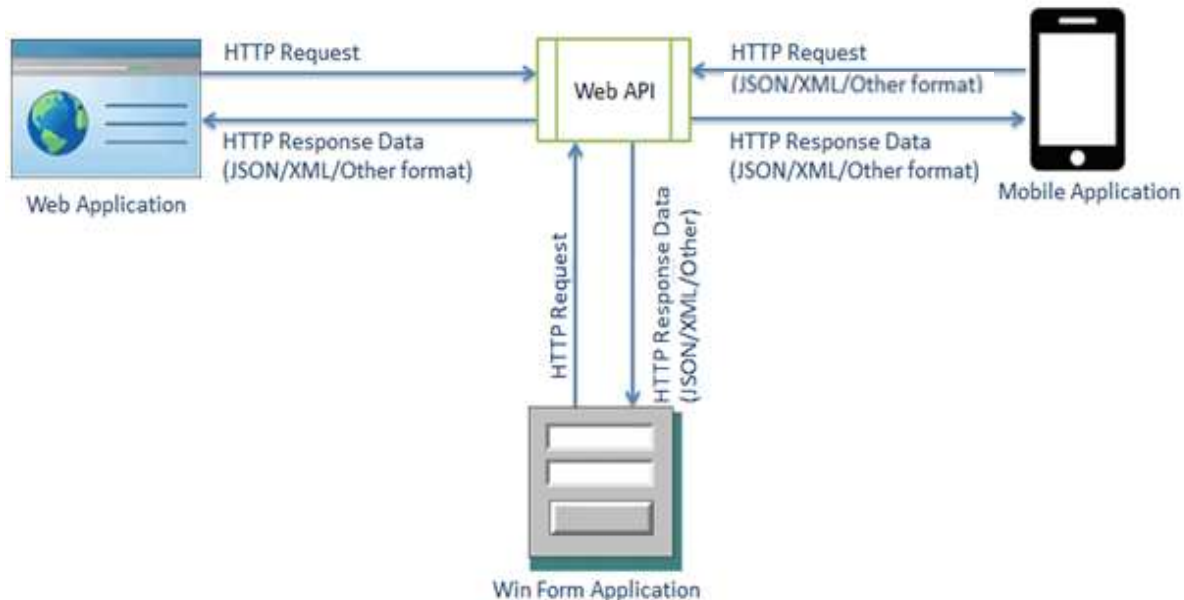


Uses of Web API

- It is used to access service data in web applications as well as many mobile apps and other external devices.
- It is used to create RESTful web services. REST stands for REpresentational State Transfer, which is an architectural style for networked hypermedia applications.
- It is primarily used to build Web Services that are lightweight, maintainable, scalable and support limited bandwidth.
- It is used to create a simple HTTP Web Service. It supports XML, JSON, and other data formats.

ASP.NET Web API

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view. It is like a web service or WCF service but the exception is that it only supports HTTP protocol.



ASP.NET Web API Characteristics

1. ASP.NET Web API is an ideal platform for building RESTful services.
2. ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
3. ASP.NET Web API maps HTTP verbs to method names.
4. ASP.NET Web API supports different formats of response data. Built-in support for formats like JSON, XML etc.
5. ASP.NET Web API can be hosted in IIS, Self-hosted or another web server that supports .NET 4.0+.
6. ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

How to use Web API

Web API receives requests from different types of client devices like mobile, laptop, etc. and then sends those requests to the webserver to process those requests and returns the desired output to the client.

Web API is a System to System interaction, in which the data or information from one system can be accessed by another system, after the completion of execution, the resultant data or we can say as output is shown to the viewer.

- Web API provides data to its programmers which is made available to outside users. When programmers decide to make some of their data available to the public, they “expose endpoints” meaning they publish a portion of the language they have used to build their program. Other programmers can then extract the data from the application by building URLs or using HTTP clients to request data from those endpoints.
- **Server Side:** A server-side web API is a programmatic interface. It consists of one or more publicly exposed endpoints. It defines a request-response message system.
- **Client Side:** Client-Side web APIs target standardized JavaScript bindings. Google created their native client architecture designed to replace native plug-ins with secure native sandboxed extensions and applications.

4.1	RESTful Web Services
------------	-----------------------------

RESTful Web Services is a lightweight, maintainable, and scalable service that is built on the REST architecture. RESTful Web Service, expose API from your application in a secure, uniform, stateless manner to the calling client. The calling client can perform predefined operations using the RESTful service. The underlying protocol for REST is HTTP.

The key elements of a RESTful implementation are as follows:

1. **Resources** – The first key element is the resource itself. Let us assume that a web application on a server has records of several employees. Let’s assume the URL of the web application is **<http://demo.guru99.com>**. Now in order to access an employee record resource via REST services, one can issue the command **<http://demo.guru99.com/employee/1>** This command tells the web server to please provide the details of the employee whose employee number is 1.
2. **Request Verbs** – These describe what you want to do with the resource. A browser issues a GET verb to instruct the endpoint it wants to get data. However, there are many other verbs available including things like POST, PUT, and DELETE. So in the case of the example **<http://demo.guru99.com/employee/1>** , the web browser is actually issuing a GET Verb because it wants to get the details of the employee record.
3. **Request Headers** – These are additional instructions sent with the request. These might define the type of response required or the authorization details.
4. **Request Body** – Data is sent with the request. Data is normally sent in the request when a POST request is made to the REST web services. In a POST call, the client actually tells the REST web services that it wants to add a

resource to the server. Hence, the request body would have the details of the resource which is required to be added to the server.

5. **Response Body** – This is the main body of the response. So in our RESTful API example, if we were to query the web server via the request **`http://demo.guru99.com/employee/1`**, the web server might return an XML document with all the details of the employee in the Response Body.
6. **Response Status Codes** – These codes are the general codes which are returned along with the response from the web server. An example is the code 200 which is normally returned if there is no error when returning a response to the client.

RESTful Methods

The below diagram shows mostly all the verbs (POST, GET, PUT, and DELETE) and an REST API example of what they would mean.

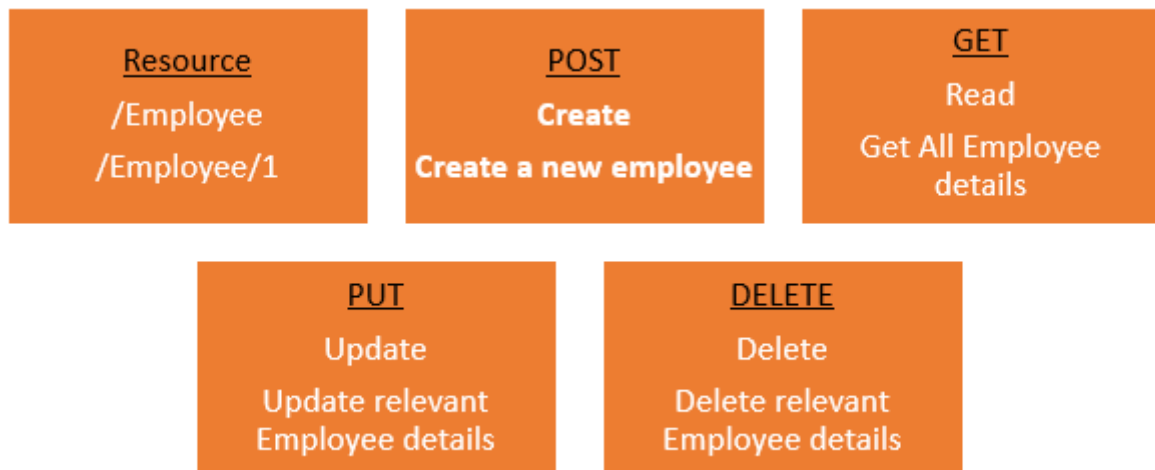
Let's assume that we have a RESTful web service which is defined at the location **`http://demo.guru99.com/employee`**. When the client makes any request to this web service, it can specify any of the normal HTTP verbs of GET, POST, DELETE and PUT. Below is what would happen If the respective verbs were sent by the client.

1. **POST** – This would be used to create a new employee using the RESTful web service
2. **GET** – This would be used to get a list of all employee using the RESTful web service
3. **PUT** – This would be used to update all employee using the RESTful web service
4. **DELETE** – This would be used to delete all employee using the RESTful services

Let's take a look from a perspective of just a single record. Let's say there was an employee record with the employee number of 1.

The following actions would have their respective meanings.

1. **POST** – This would not be applicable since we are fetching data of employee 1 which is already created.
2. **GET** – This would be used to get the details of the employee with Employee no as 1 using the RESTful web service
3. **PUT** – This would be used to update the details of the employee with Employee no as 1 using the RESTful web service
4. **DELETE** – This is used to delete the details of the employee with Employee no as 1



Why RESTful?

RESTful mostly came into popularity due to the following reasons:

1. Heterogeneous Languages and Environments – This is one of the fundamental reasons which is the same as we have seen for [SOAP](#) as well.

- It enables web applications that are built on various programming languages to communicate with each other
- With the help of RESTful services, these web applications can reside on different environments, some could be on Windows, and others could be on Linux.

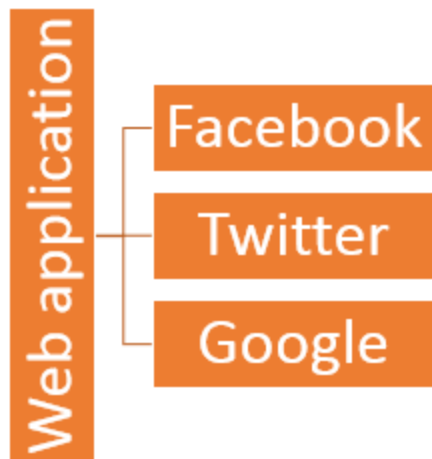
But in the end, no matter what the environment is, the end result should always be the same that they should be able to talk to each other. RESTful web services offer this flexibility to applications built on various programming languages and platforms to talk to each other.

The below picture gives an example of a web application which has a requirement to talk to other applications such as Facebook, Twitter, and Google.

Now if a client application had to work with sites such as Facebook, Twitter, etc. they would probably have to know what is the language Facebook, Google and Twitter are built on, and also on what platform they are built on.

Based on this, we can write the interfacing code for our web application, but this could prove to be a nightmare.

Facebook, Twitter, and Google expose their functionality in the form of RESTful web services. This allows any client application to call these web services via REST.



2. The event of Devices – Nowadays, everything needs to work on [Mobile](#) devices, whether it be the mobile device, the notebooks, or even car systems.

Can you imagine the amount of effort to try and code applications on these devices to talk with normal web applications? Again RESTful API's can make this job simpler because as mentioned in point no 1, you really don't need to know what is the underlying layer for the device.

3. Finally in the event of the Cloud – Everything is moving to the cloud. Applications are slowly moving to cloud-based systems such as in Azure or Amazon. Azure and Amazon provide a lot of API's based on the RESTful architecture. Hence, applications now need to be developed in such a way that they are made compatible with the Cloud. So since all Cloud-based architectures work on the REST principle, it makes more sense for web services to be programmed on the REST services based architecture to make the best use of Cloud-based services.

RESTful Architecture

An application or architecture considered RESTful or REST-style has the following characteristics

1. State and functionality are divided into distributed resources – This means that every resource should be accessible via the normal HTTP commands of GET, POST, PUT, or DELETE. So if someone wanted to get a file from a server, they should be able to issue the GET request and get the file. If they want to put a file on the server, they should be able to either issue the POST or PUT request. And finally, if they wanted to delete a file from the server, they should issue the DELETE request.

2. The architecture is client/server, stateless, layered, and supports caching

- Client-server is the typical architecture where the server can be the web server hosting the application, and the client can be as simple as the web browser.
- Stateless means that the state of the application is not maintained in REST. For example, if you delete a resource from a server using the DELETE command, you cannot expect that delete information to be passed to the next request.

In order to ensure that the resource is deleted, you would need to issue the GET request. The GET request would be used to first get all the resources on the server. After which one would need to see if the resource was actually deleted.

RESTful Principles and Constraints

The REST architecture is based on a few characteristics which are elaborated below. Any RESTful web service has to comply with the below characteristics in order for it to be called RESTful. These characteristics are also known as design principles which need to be followed when working with RESTful based services.

1. RESTful Client-Server



This is the most fundamental requirement of a REST based architecture. It means that the server will have a RESTful web service which would provide the required functionality to the client. The client sends a request to the web service on the server. The server would either reject the request or comply and provide an adequate response to the client.

2. Stateless

The concept of stateless means that it's up to the client to ensure that all the required information is provided to the server. This is required so that server can

process the response appropriately. The server should not maintain any sort of information between requests from the client. It's a very simple independent question-answer sequence. The client asks a question, the server answers it appropriately. The client will ask another question. The server will not remember the previous question-answer scenario and will need to answer the new question independently.

3. Cache



The Cache concept is to help with the problem of stateless which was described in the last point. Since each server client request is independent in nature, sometimes the client might ask the server for the same request again. This is even though it had already asked for it in the past. This request will go to the server, and the server will give a response. This increases the traffic across the network. The cache is a concept implemented on the client to store requests which have already been sent to the server. So if the same request is given by the client, instead of going to the server, it would go to the cache and get the required information. This saves the amount of to and fro network traffic from the client to the server.

4. Layered System

The concept of a layered system is that any additional layer such as a middleware layer can be inserted between the client and the actual server hosting the RESTful web service (The middleware layer is where all the business logic is created. This can be an extra service created with which the client could interact with before it makes a call to the web service.). But the introduction of this layer needs to be transparent so that it does not disturb the interaction between the client and the server.

5. Interface/Uniform Contract

This is the underlying technique of how RESTful web services should work. RESTful basically works on the HTTP web layer and uses the below key verbs to work with resources on the server

- POST – To create a resource on the server
- GET – To retrieve a resource from the server

- PUT – To change the state of a resource or to update it
- DELETE – To remove or delete a resource from the server

Create your first RESTful web service in ASP.NET

Now in this REST API tutorial, we will learn how to create a RESTful web service in ASP.NET:

Web services can be created in a variety of languages. Many Integrated Development Environments can be used to create REST-based services.

In this RESTful API example, we are going to create our REST application in .Net using Visual Studio. In our example, for RESTful web services we are going to emulate the following REST service example.

We are going to have a RESTful web service which will work on the below set of data.

The below set of data represents an REST API example of having a company which exposes the Tutorial's they have based on the Tutorialid.

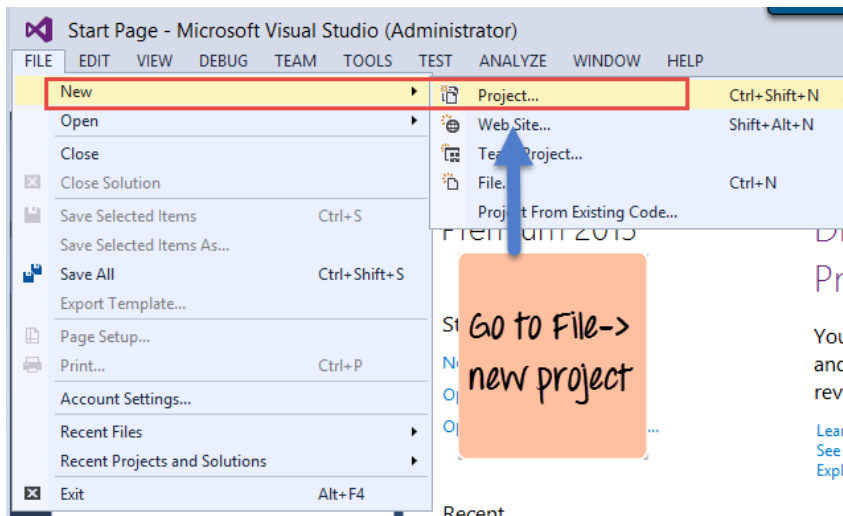
Tutorialid	TutorialName
0	Arrays
1	Queues
2	Stacks

In our REST API tutorial example, we are going to implement the below RESTful Verbs.

1. **GET Tutorial** – When a client invokes this RESTful API, they will be given the entire set of Tutorials available from the web service.
2. **GET Tutorial/Tutorialid** – When a client invokes this RESTful API, they will be given the Tutorial name based on the Tutorialid sent by the client.
3. **POST Tutorial/Tutorialname** – When a client invokes this RESTful API, the client will submit a request to insert a Tutorialname. The web service will then add the submitted Tutorial name to the collection.
4. **DELETE Tutorial/Tutorialid** – When a client invokes this RESTful API, the client will submit a request to delete a Tutorialname based on the Tutorialid. The web service will then delete the submitted Tutorial name from the collection.

Let's follow the below steps in this RESTful API tutorial to create our first RESTful web services, which carries out the above implementation.

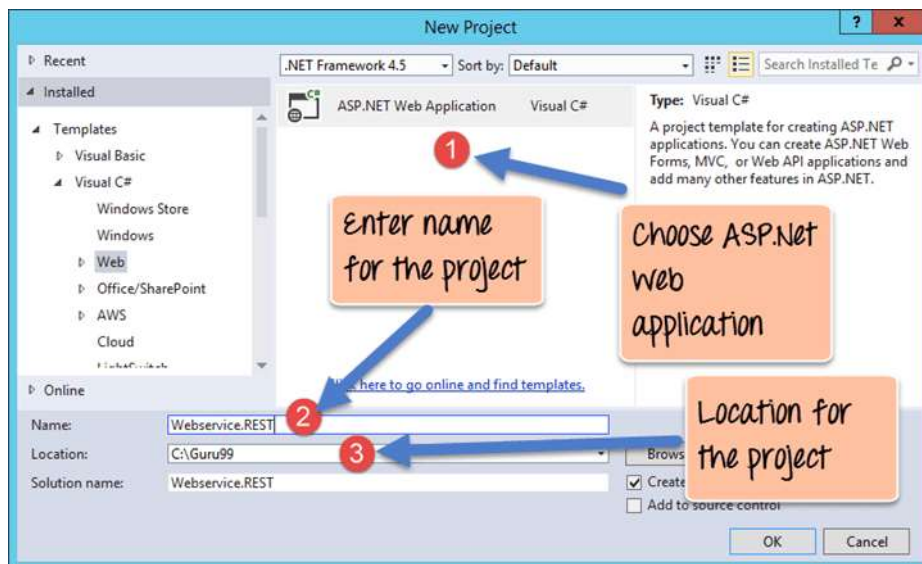
Step 1) The first step is to create an empty Asp.Net Web application. From Visual Studio 2013, click on the menu option File->New project.



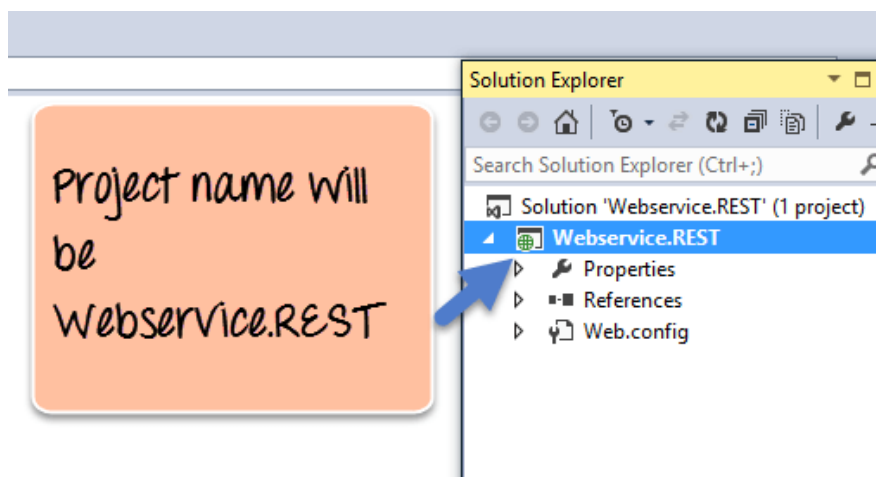
Once you click on the New Project option, Visual Studio will then give you another dialog box for choosing the type of project and to give the necessary details of the project. This is explained in the next step of this RESTful API tutorial

Step 2) In this step,

1. Ensure to first choose the RESTful web services C# web template of ASP.NET Web application. The project has to be of this type in order to create web services project. By choosing this options, Visual Studio will then carry out the necessary steps to add required files which are required by any web-based application.
2. Give a name for your project which in our case has been given as "Webservice.REST".
3. Then ensure to give a location, where the project files will be stored.

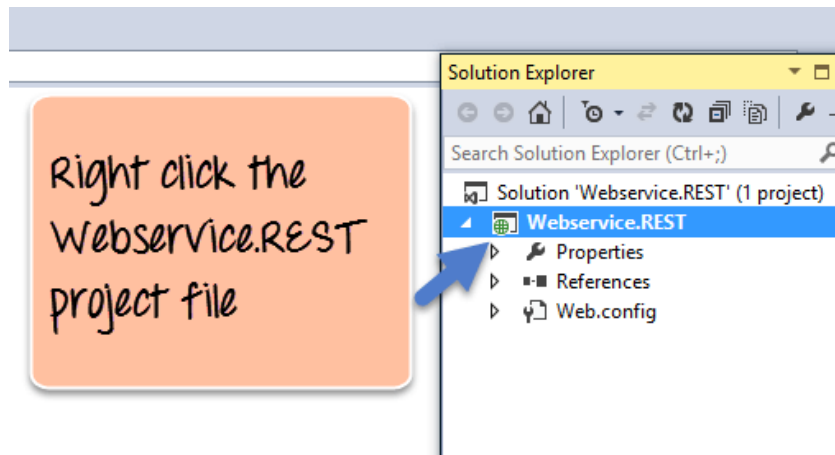


Once done you will see the project file created in your solution explorer in Visual Studio 2013.

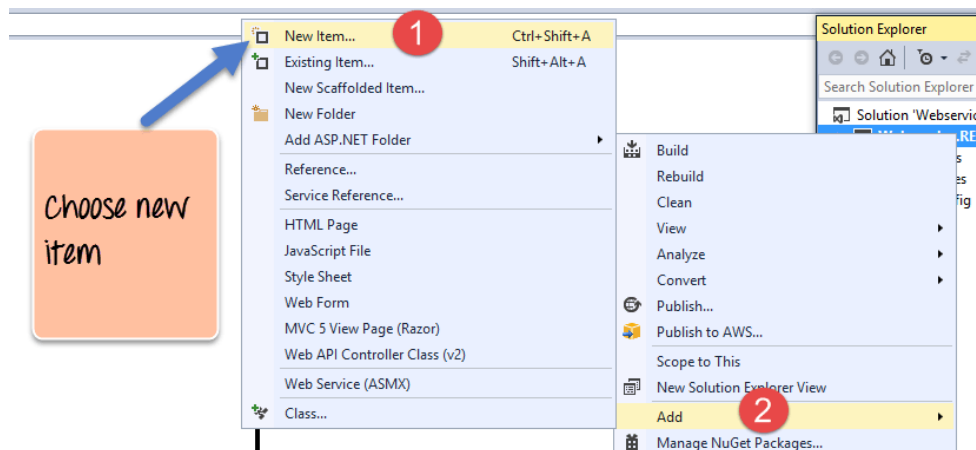


Step 3) The next step is to create the web service file which is going to have the RESTful web service

1. First Right-click on the project file as shown below

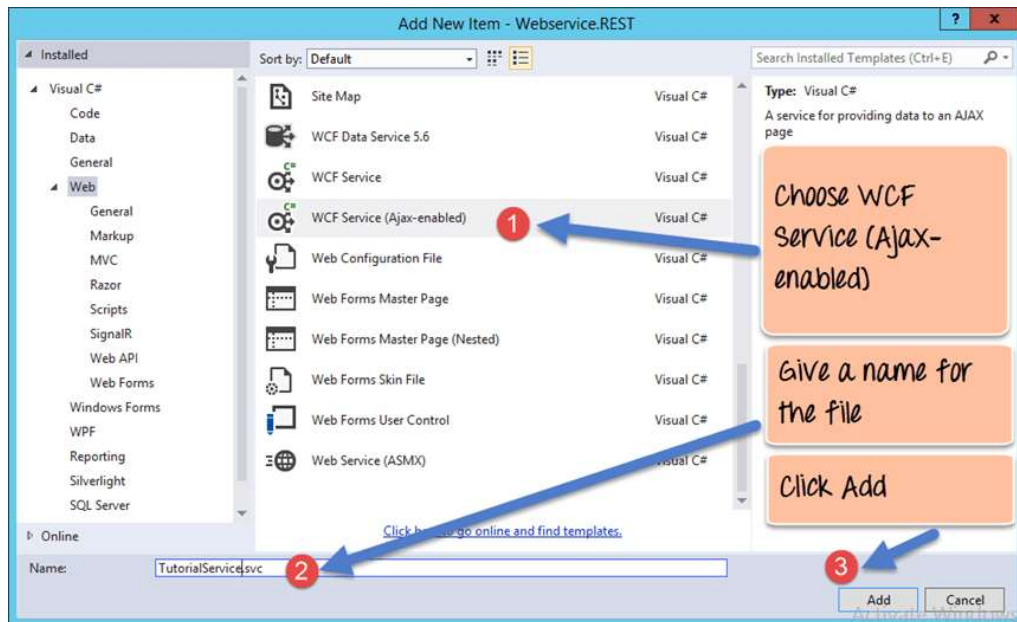


2. In this step,
1. Right-click on the project file
2. Choose the option "Add->new item."



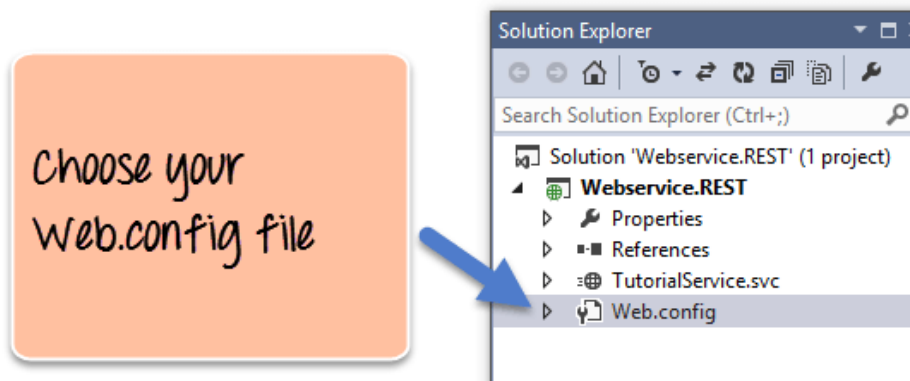
In the dialog box which appears, you need to perform the following

1. Choose the option of WCF Service (Ajax-enabled) – Choose a file of this type, it causes the Visual studio to add some basic code which helps one create a RESTful web service. WCF stands for Windows Communication Foundation. WCF is a library for applications of various platforms or the same platform to communicate over the various protocols such as TCP, HTTP, HTTPS. Ajax basically is Asynchronous JavaScript and XML. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes.
2. Next give a name for the service which is TutorialService in our case.
3. Finally, click the Add button to add the service to the solution.



Step 4) The next step is to actually make a configuration change to enable this project to complete the work with RESTful web services. This requires to make a change to the file called **Web.config**. This file appears in the same window as the Webservice project file. The file Web.config contains all configurations that make the web application to work as it should. The change being made actually allows the application to send and receive data as a pure RESTful web service.

1. Click on the Web.config file to open the code



2. Find for the line `<enableWebScript>`


```

</system.web>
<system.serviceModel>
  <behaviors>
    <endpointBehaviors>
      <behavior name="Webservice.REST.TutorialServiceAspNetAjaxBehavior">

        <enableWebScript />

      </behavior>
    </endpointBehaviors>
  </behaviors>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
    multipleSiteBindingsEnabled="true" />
</services>
<service name="Webservice.REST.TutorialService">

```

Find for enableWebScript

3. Change the line to <webHttp>

```

</system.web>
<system.serviceModel>
  <behaviors>
    <endpointBehaviors>
      <behavior name="Webservice.REST.TutorialServiceAspNetAjaxBehavior">

        <webHttp />

      </behavior>
    </endpointBehaviors>
  </behaviors>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
    multipleSiteBindingsEnabled="true" />
</services>
<service name="Webservice.REST.TutorialService">

```

Change to webHttp

Step 5) The next step in this RESTful API tutorial is to add our code for implementation. All of the below-mentioned code has to be written in the TutorialService.svc file

1. The first bit is to add code to represent our data which will be used in our program. So we are going to have a list of string variables with values of "Arrays", "Queues" and "Stacks". This will represent the tutorials name available through our hosting web service.

```

namespace Webservice.REST
{
    [ServiceContract(Namespace = "")]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
    public class TutorialService
    {
        private static List<String> lst = new List<String>
            (new String[] { "Arrays", "Queues", "Stacks" });
    }
}

```

Define the list if strings

```

namespace Webservice.REST
{

```

```

    [ServiceContract(Namespace = "")]
    [AspNetCompatibilityRequirements(RequirementsMode
    AspNetCompatibilityRequirementsMode.Allowed
    =

```

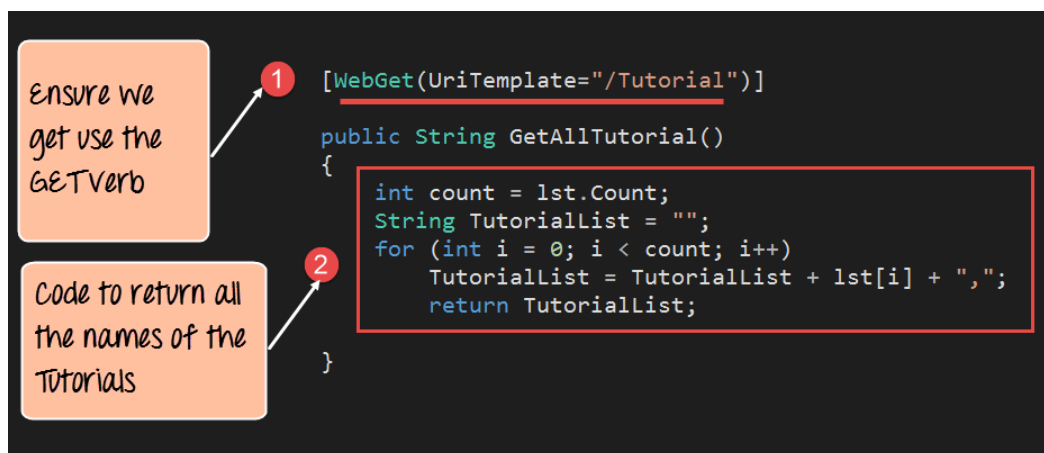


```
public class TutorialService
{
    private static List<String> lst = new List<String>
    (new String[] {"Arrays","Queues","Stacks"});
```

Step 6) Next we will define the code for our GET method. This code will also reside in the same TutorialService.svc file. This code will run whenever we call the service from our browser.

The below method will be used to fulfil the desired scenario

- If a user wants a list of all Tutorials available, then the below code would need to be written to accomplish this.



```
[WebGet(UriTemplate="/Tutorial")]
```

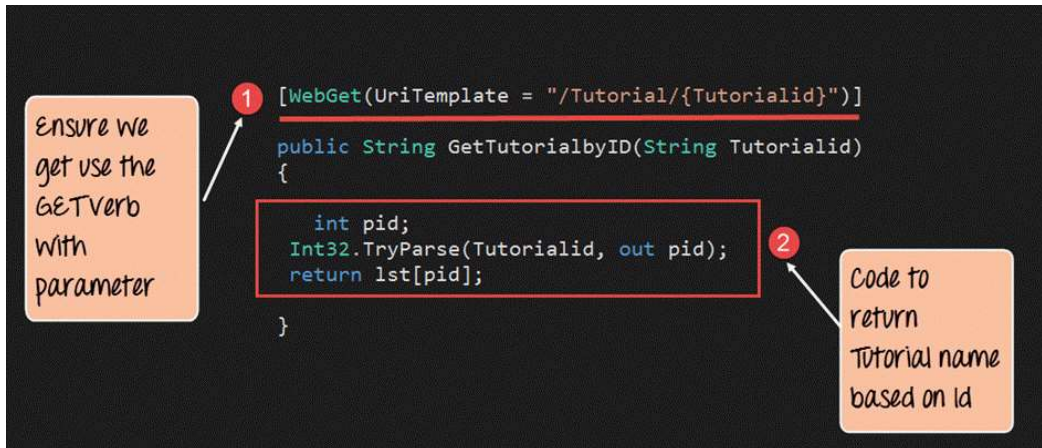
```
public String GetAllTutorial()
{
    int count = lst.Count;
    String TutorialList = "";
    for (int i = 0; i < count; i++)
        TutorialList = TutorialList + lst[i] + ",";
    return TutorialList;
}
```

Code Explanation:

1. The first line of code is the most important. It is used to define how we can call this method via a URL. So if the link to our web service is ***http://localhost:52645/TutorialService.svc*** and if we append the `'/Tutorial'` to the URL as ***http://localhost:52645/TutorialService.svc/Tutorial***, the above code will be invoked. The attribute of 'WebGet' is a parameter which allows this method to be a RESTful method so that it can be invoked via the GET verb.

2. This section of code is used to go through our list of strings in the 'lst' variable and return all of them to the calling program.

Step 7) The code below ensures that if a GET call is made to the Tutorial Service with a Tutorial id, then it would return the corresponding Tutorial Name based on the Tutorial id.



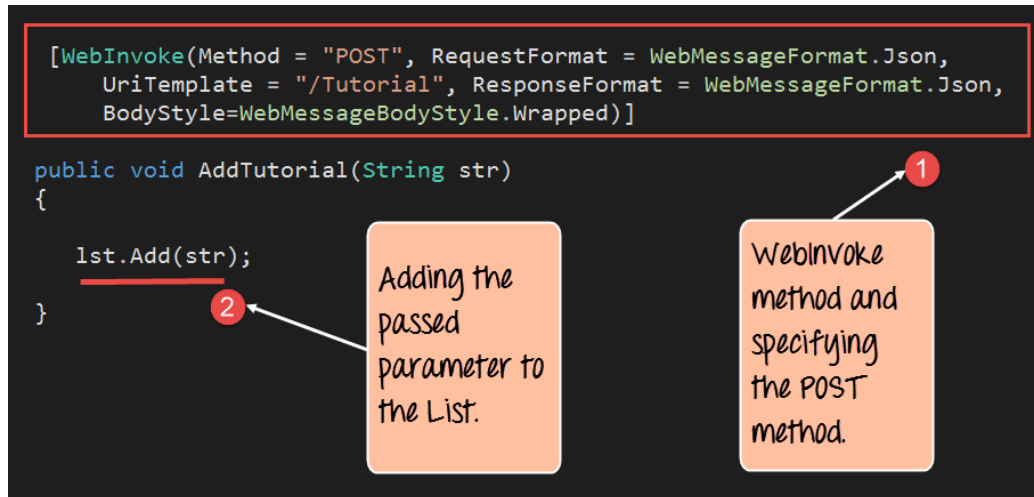
```
[WebGet (UriTemplate = "/Tutorial/{Tutorialid}")]
```

```
public String GetTutorialbyID(String Tutorialid)
{
    int pid;
    Int32.TryParse(Tutorialid, out pid);
    return lst[pid];
}
```

Code Explanation:

1. The first line of code is the most important. It is used to define how we can call this method via a URL. So if the link to our web service is **http://localhost:52645/TutorialService.svc** and if we append the '/Tutorial/{Tutorialid}' to the URL, then we would be able to call the web service as **http://localhost:52645/TutorialService.svc/Tutorial/1** as an example. The web service would then need to return the Tutorial name which had the Tutorial id#1.
2. This section of code is used to return the "Tutorial name" which has the Tutorial id passed to the web method.
 - By default, what needs to be remembered is that whatever is passed to the URL in the browser is a string.
 - But you have to remember that the Index to our list has to be an integer, so we are adding the necessary code to first convert the Tutorialid to an Integer and then use it to access the index position in our list and
 - Then return the value to the calling program accordingly.

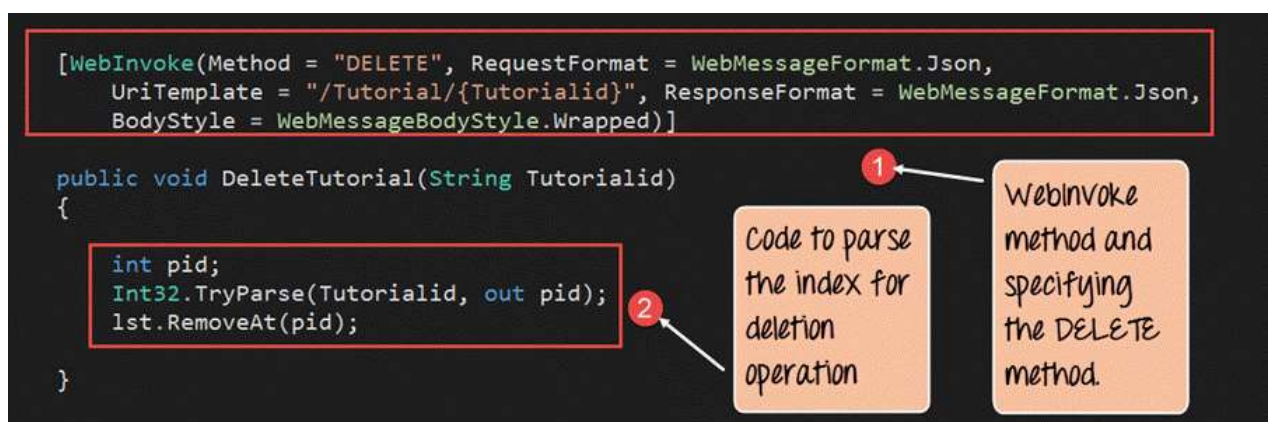
Step 8) The next step is to write up the code for our POST method. This method will be invoked whenever we want to add a string value to our list of Tutorials via the POST method. For example, if you wanted to add the Tutorial name of "Software Testing" then you would need to use the POST method.



Code Explanation:

1. The first line is the 'WebInvoke' attribute which has been attached to our method. This allows the method to be invoked via the POST call. The RequestFormat and ResponseFormat attribute have to be mentioned as JSON, since when posting values to a RESTful web service, the values have to be in this format.
2. The second line of code is used to add the string value passed via the POST call to our existing list of Tutorial strings.

Step 9) Finally we are going to add our method to handle the DELETE operation. This method will be invoked whenever we want to delete an existing string value from our list of Tutorials via the DELETE method.



[WebInvoke(Method = "DELETE", RequestFormat = WebMessageFormat.Json,

```

        UriTemplate = "/Tutorial/{Tutorialid}", ResponseFormat =
WebMessageFormat.Json,
        BodyStyle = WebMessageBodyStyle.Wrapped)]

public void DeleteTutorial(String Tutorialid)
{
    int pid;
    Int32.TryParse(Tutorialid, out pid);
    1st.RemoveAt(pid);
}

```

Code Explanation:

1. The first line is the 'WebInvoke' attribute which has been attached to our method. This allows the method to be invoked via the POST call. The RequestFormat and ResponseFormat attribute have to be mentioned as JSON, since when posting values to a RESTful web service, the values have to be in this format. Note that the Method parameter is being set to "DELETE." This means that whenever we issue the DELETE verb, this method will be invoked.
2. The second line of code is used to take the Tutorialid sent via the DELETE call and subsequently delete that id from our list. (The **Int32** function in code is used to convert the Tutorial ID from a string variable to an integer).

4.2	API Security: Best Practices
------------	-------------------------------------

Securing your API against the attacks should be based on:

Authentication – Determining the identity of an end user. In a REST API, basic authentication can be implemented using the TLS protocol, but OAuth 2 and OpenID Connect are more secure alternatives.

Authorization – Determining the resources an identified user can access. An API should be built and tested to prevent users from accessing API functions or operations outside their predefined role. For example, a read-only API client shouldn't be allowed to access an endpoint providing admin functionality.

Maintaining Session - RESTful services work on a stateless protocol, in other words HTTP. We can maintain sessions in the Web API using token-based authorization techniques. An authenticated user will be allowed to access resources for a specific period of time and can re-instantiate the request with an increased session time delta to access other resource or the same resource. Websites using WebAPIs as RESTful services may need to implement login/logout for a user, to maintain sessions for the user, to provide roles and permissions to

their user, all these features could be done using basic authentication and token-based authorization.

Integrity, Non-Repudiation-Making sure that a message remains unaltered during transit by having the sender digitally sign the message. A digital signature is used to validate the signature and provides non-repudiation. The timestamp in the signature prevents anyone from replaying this message after the expiration.

Basic Authentication

Basic authentication is a mechanism, where an end-user is authenticated using our service, in other words RESTful service, using plain credentials such as user name and password. An end-user makes a request to the service for authentication with the user name and password embedded in the request header. The service receives the request and checks if the credentials are valid or not and returns the response accordingly, in case of invalid credentials, the service responds with a 401-error code, in other words unauthorized. The actual credentials by which the comparison is done may lie in the database, any config file like web.config or in the code itself.

- **Pros and Cons of Basic Authentication**

Basic authentication has its own pros and cons. It is advantageous for implementation, it is very easy to implement, it is supported by nearly all the modern browsers and has become an authentication standard in RESTful / Web APIs. It has the disadvantages of sending user credentials in plain text, sending user credentials inside a request header, in other words prone to hack. One must send credentials each time a service is called. No session is maintained and a user cannot logout once logged in using basic authentication. It is very prone to Cross-Site Request Forgery (CSRF).

Token Base Authorization

The authorization part comes just after authentication. Once authenticated, a service can send a token to an end-user by which the user can access other resources. The token could be any encrypted key that only the server/service understands and when it fetches the token from the request made by the end user, it validates the token and authorizes the user into the system. The token generated could be stored in a database or an external file as well, in other words we need to persist the token for future references. The token can have its own lifetime and may expire accordingly. In that case the user will need to be authenticated again into the system.

Additional best practices include validating your API calls against API schemas that clearly describe expected structures. Scanning payloads and performing schema

validation can prevent code injections, malicious entity declarations and parser attacks. Assigning an API token for each API call validates incoming queries and prevents attacks on endpoints.

Security Modes

Programming WCF security entails a few critical decision points. One of the most basic is the choice of *security mode*. The two major security modes are *transport mode* and *message mode*.

A third mode, which combines the semantics of both major modes, is transport with message credentials mode.

The security mode determines how messages are secured, and each choice has advantages and disadvantages, as explained below.

Transport Mode

There are several layers between the network and the application. One of these is the *transport* layer, which manages the transfer of messages between endpoints. For the present purpose, it is only required that you understand that WCF uses several transport protocols, each of which can secure the transfer of messages.

A commonly used protocol is HTTP; another is TCP. Each of these protocols can secure message transfer by a mechanism (or mechanisms) particular to the protocol. For example, the HTTP protocol is secured using SSL over HTTP, commonly abbreviated as "HTTPS." Thus, when you select the transport mode for security, you are choosing to use the mechanism as dictated by the protocol. For example, if you select the `WSHttpBinding` class and set its security mode to Transport, you are selecting SSL over HTTP (HTTPS) as the security mechanism. The advantage of the transport mode is that it is more efficient than message mode because security is integrated at a comparatively low level. When using transport mode, the security mechanism must be implemented according to the specification for the transport, and thus messages can flow securely only from point-to-point over the transport.

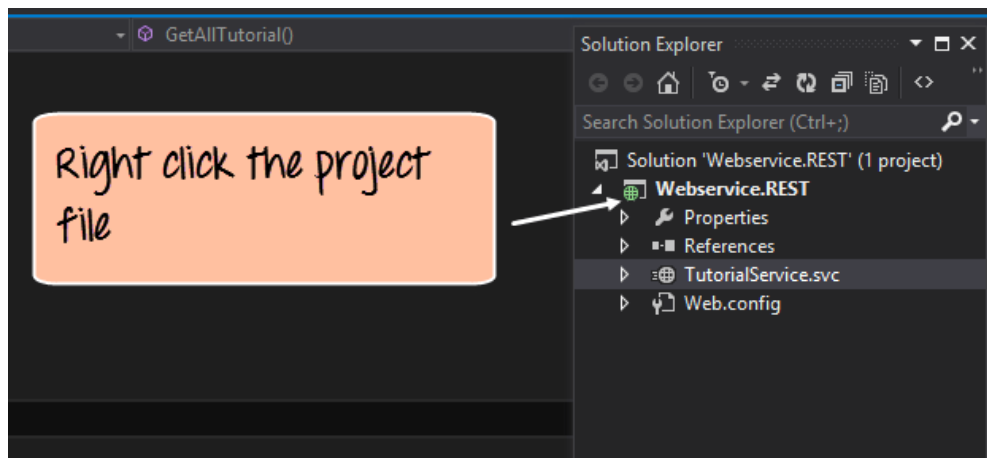
Message Mode

In contrast, message mode provides security by including the security data as part of every message. Using XML and SOAP security headers, the credentials and other data needed to ensure the integrity and confidentiality of the message are included with every message.

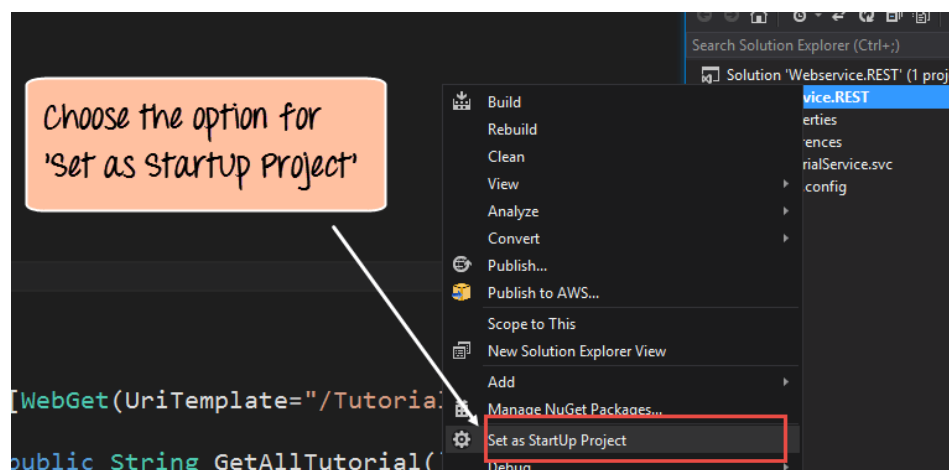
Every message includes security data, so it results in a toll on performance because each message must be individually processed. (In transport mode, once the transport layer is secured, all messages flow freely.) However, message security has one advantage over transport security: it is more flexible. That is, the security requirements are not determined by the transport. You can use any type of client credential to secure the message. (In transport mode, the transport protocol determines the type of client credential that you can use.)

To run the web service, please follow the below steps

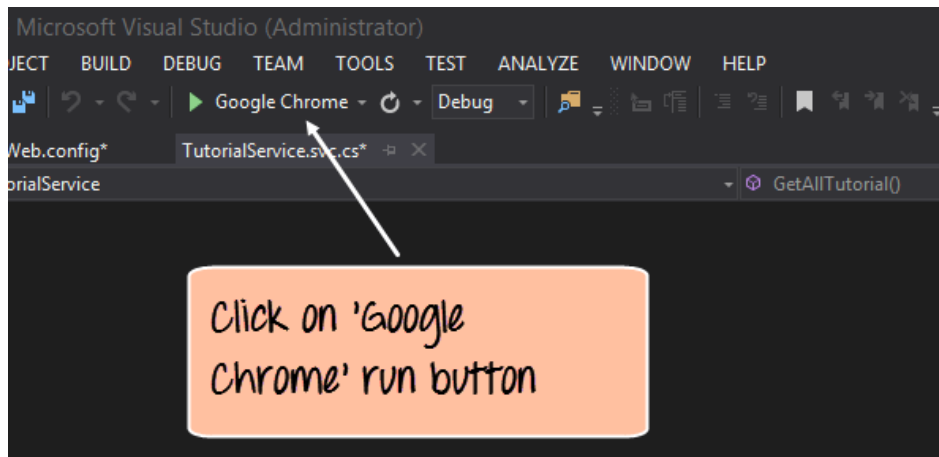
Step 1) Right click on the Project file – Webservice.REST



Step 2) Choose the menu option 'Set as StartUp Project'. This will ensure that this project is run when Visual Studio runs the entire solution

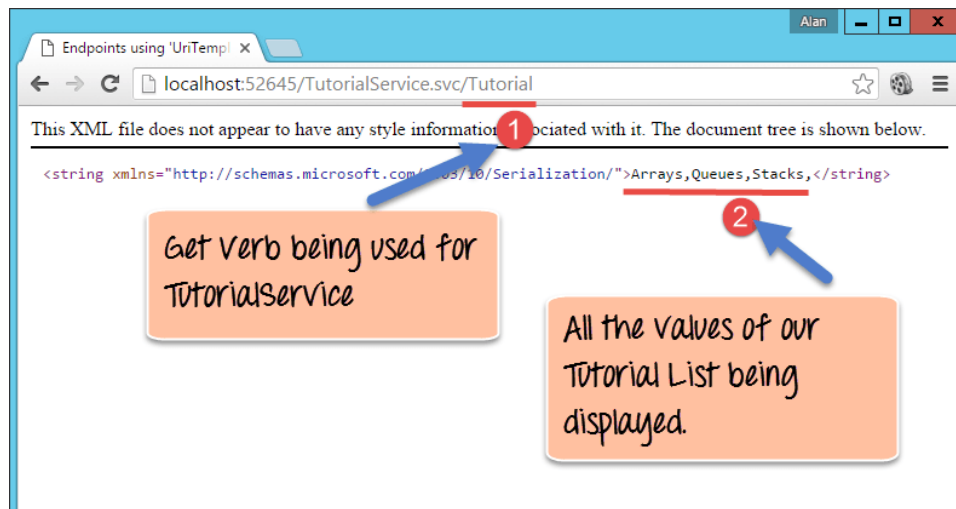


Step 3) The next step is to run the project itself. Now depending on the default browser installed on the system, the appropriate browser name will come next to the run button in Visual Studio. In our case, we have Google Chrome showing up. Just click on this button.



Output:

When the project is run, you can browse to your TutorialService.svc/Tutorial section, and you will get the below output.



In the above output,

- You can see that the browser is invoking the 'GET' verb and executing the 'GetAllTutorial' method in the web service. This module is used to display all the Tutorials exposed by our web service.

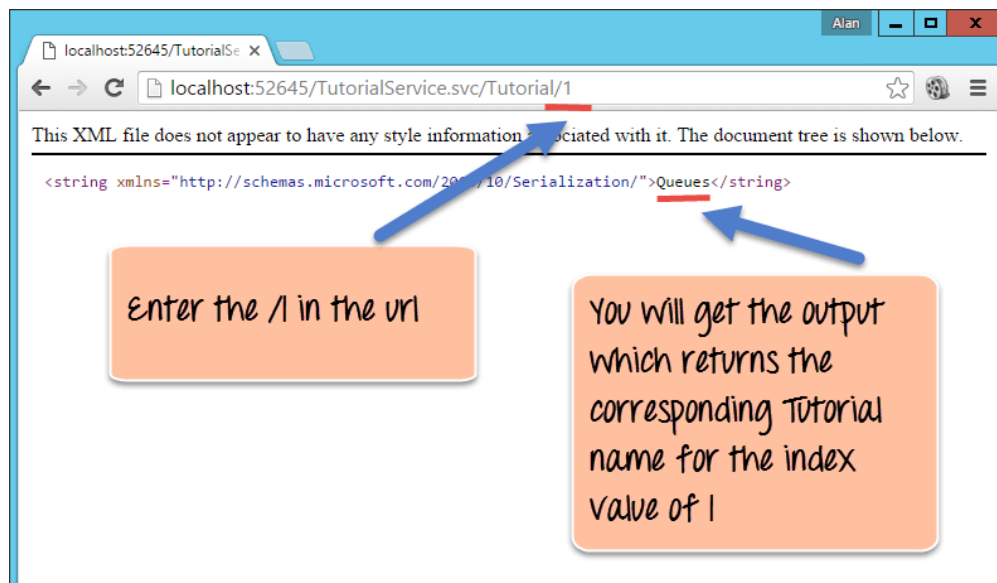
Testing your first RESTful web service

In the above section, we have already seen how to use the browser to execute the 'GET' verb and invoke the 'GetAllTutorial.'

- Let's now use the browser to execute the following use case scenario.

GET Tutorial/Tutorialid – When a client invokes this RESTful API, they will be given the Tutorial name based on the Tutorialid sent by the client

In your browser, append the string /1 after the Tutorial word in the URL. If you hit the enter button, you will get the below output



Now you will see the output of Queues which actually corresponds to the number 1 in our list of Tutorial Strings. This means that the 'GetTutorialbyID' method is now being invoked from our Webservice. It also shows that the value of 1 is being passed successfully via the browser to our web service and to our method and that is why we are getting the correct corresponding value of "Queues" in the browser.

2. Next let's consume our web service by executing the below scenario. For this, you need to install the tool called "Fiddler" which is a free downloadable tool from the site.

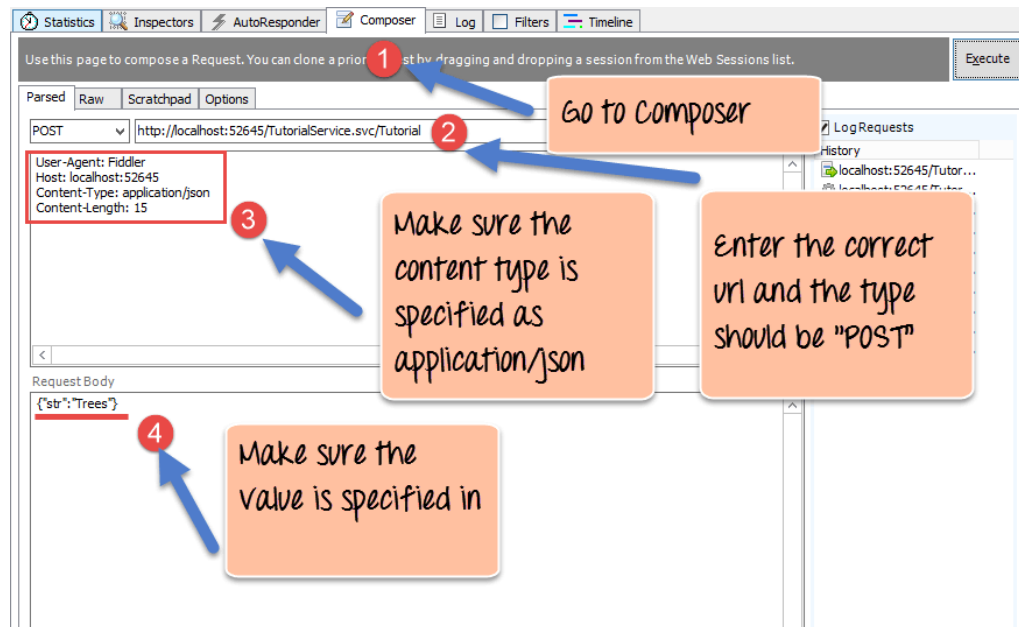
POST Tutorial/Tutorialname – When a client invokes this RESTful API, the client will submit a request to insert a Tutorialname. The web service will then add the submitted Tutorial name to the collection.

Run the Fiddler tool and perform the below steps;

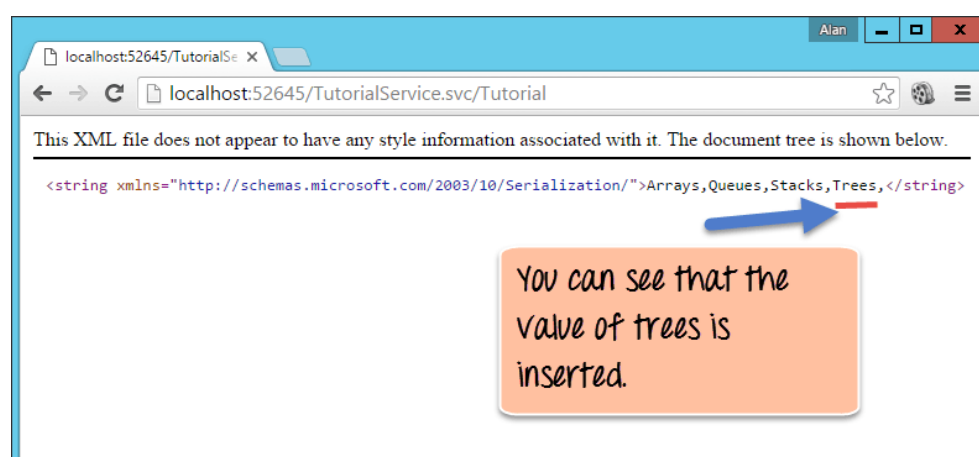
1. Go to the composer section. This is used to create requests which can be submitted to any webapplication.
2. Make sure the request type is "POST" and the correct URL is being hit, which in our case should be
http://localhost:52645/TutorialService.svc/Tutorial
3. Make sure the Content-Type is marked as application/json. Remember that our POST request method in our Web service only accepts json style data so we need to ensure this is specified when we are sending a request to our application.

4. Finally, we need to enter our data. Remember that our method for POST accepts a parameter called 'str'. So here we are specifying that we want to add a value called "Trees" to our collection of Tutorial names and ensure that it is tagged to the str variable name.

Finally, just click the Execute button in fiddler. This will send a request to the web service to POST the data "Trees" to our web service.



Now, when we browse to the Tutorial URL to show all the strings in our Tutorial list, you will now see the value of "Trees" is also present. This shows that the POST request to the web service was successfully executed and that it was successfully added to our Tutorial List.



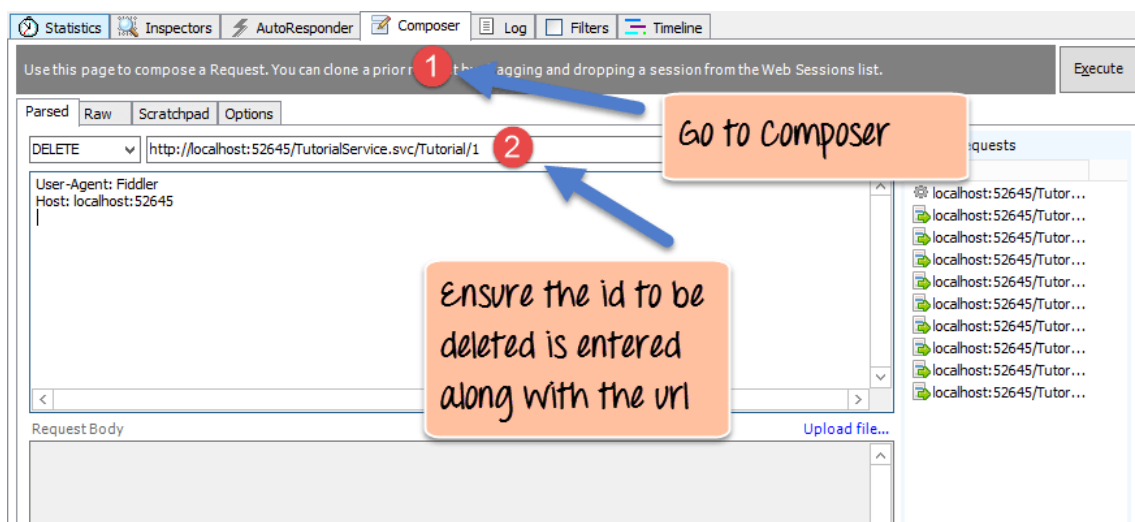
3. Next let's consume our web service by executing the below scenario. For this also we need to use the fiddler tool

DELETE Tutorial/Tutorialid- When a client invokes this RESTful API, the client will submit a request to delete a Tutorialname based on the Tutorialid. The web service will then delete the submitted Tutorial name from the collection.

Run the Fiddler tool and perform the below steps

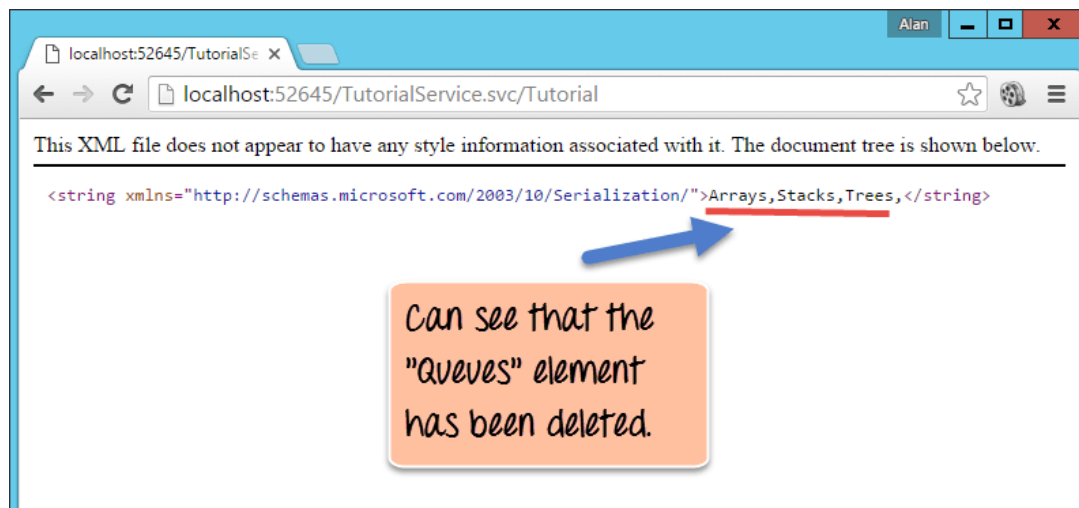
1. Go to the composer section. This is used to create requests which can be submitted to any webapplication.
2. Make sure the request type is "DELETE" and the correct URL is being hit, which in our case should be
http://localhost:52645/TutorialService.svc/Tutorial. Ensure that the id which is used to delete a string in the list sent via the URL as a parameter. In our REST example, we are sending 1 so this will delete the 2nd element in our collection which is "Queues".

Finally, just click the Execute button in fiddler. This will send a request to the web service to DELETE the data "Queues" to our web service.



Now, when we browse to the Tutorial URL to show all the strings in our Tutorial list, you will notice that the value of "Queues" is no longer present.

This shows that the DELETE request to the web service was successfully executed. The element at index no 1 in our list of Tutorial strings was successfully deleted.



Summary

- REST stands for Representational State Transfer. REST is used to build web services that are lightweight, maintainable, and scalable in nature.
- More and more applications are moving to the RESTful architecture. This is because of the fact that there are a lot of people now using mobile devices and a wider variety of applications moving to the cloud.
- The main aspects of REST are the resources which reside on the server and the verbs of GET, POST, PUT and DELETE, which can be used to work with these resources.
- Visual Studio and .Net can be used to create RESTful web services.
- When Testing web services for POST and PUT, you need to use another tool called fiddler which can be used to send the POST and PUT request to the server.

5.0 Securing Mobile Apps

Security of web-applications and mobile apps is very important due to diverse Development Frameworks/Operating Systems. Sometimes, the need to quickly develop mobile apps results in compromising security aspects. Android, iOS and hybrid apps are vulnerable to a range of threats, and users need to be protected from the risk associated with running mobile apps in an unprotected environment. This becomes more important when it is about the Apps for Government. Platform-specific security best practices must be followed for robust and secure mobile application development.

A smartphone user is exposed to various threats when they use their phone. Attackers exploit weakness inherent in smartphones or flaws in the application logic. Mobile apps are often the cause of unintentional mass data leakage. Client-side vulnerabilities can be exploited without physical access to the phone. Improper platform usage has been the leading mobile security vulnerability, which refers to the misuse of any platform-specific feature or failure to incorporate platform security controls. Mobile applications developed with default configurations are vulnerable to certain known attacks. It is important to know such vulnerabilities while developing mobile apps. Another most common vulnerability is insecure data storage. If an attacker can physically access the phone, the attacker can copy application data to a computer.

A mobile application resides completely in the user device. There are free tools available to decompile and regenerate source codes of a mobile application. An attacker can learn the business logic from the decompiled source code and can attack the IT infrastructure using the credentials taken from the application installed in the user device. Securely storing the secrets in a device is also very important for securing the IT infrastructure from various attacks. Platform-specific security best practices must be followed for robust and secure mobile application development.



Since the call is going to back end from the mobile application thorough API for retrieving data, proper security control must be placed in the backend. For the backend, OWASP TOP 10 should be followed and security best practices must be followed for securing entire stack including OS, database, application server , web server etc. All the communication from mobile app to back end must happen over latest TLS with secure configurations. SSL pinning may also be used to restrict the servers with which mobile applications may communicate.

5.1	Developer Challenges
------------	-----------------------------

Along with security, the privacy of user information is the primary concern for an app developer. A hacker/ intruder should be prevented from getting access to the critical data of the mobile application. The app must protect the following from a hacker/ intruder:

- Private, sensitive, and personal information
- Unauthorized access to the system
- Execution in a rooted/ jailbroken environment

A malicious user or hacker can use a rooted/ jailbroken device to install the application to study the logic and API information. The attacker can even create malicious/fake apps targeting the APIs.

Mobile applications can be login based or without login, based on the usage and security of services provided by the enterprise. Login based applications can have the following authentication mechanisms in place, with one or more factor for authentication (specifically, what user knows, has, and is), to determine the user's identify.

- Login with a username and password
- LDAP authentication
- Login using a unique id provided by the enterprise (e.g. Aadhar) along with two-factor authentication/ N-Factor (OTP, Biometrics etc.)

5.2	Security Issues - Mobile App Development
------------	---

To address the security concerns in mobile app development, following are the major challenges:

API security in Mobile App Development

It is the measure and means of defending mobile applications from digital fraud in the form of malware, hacking, and other criminal manipulation. The OWASP Mobile Security Top 10 (<https://www.owasp.org>) may be referred for more awareness about the current mobile security issues.

API security refers to the collaborative efforts directed towards safeguarding API's integrity, regardless of them being owned or used. API security encompasses everything spanning from, API access control to privacy, as well as the threat detection and remediation on API's through API reverse engineering. The phenomenon is centred around securing the application layer to directly address any malicious hacker's possibilities interacting with the API. In case the API connects to any third-party application, all data is ultimately funnelled back to the internet. API security includes network security concepts including rate-limiting or throttling, along with related concepts from identity-based security, data security,

and information monitoring/analytics. API's expose sensitive financial, medical, and personal information, which can cost hugely, both in terms of finances and reputation. So the developer need to first identify the trusted libraries/APIs for mobile app development which are framework dependent.

Storing Secrets

There should be a minimum-security mechanism in place for APIs used by remote non-sandboxed clients like mobile applications. Security of the APIs is partially dependent on the secure integration by the app. Different schemes can be in place to design a secured architecture. The factors affecting the security scheme depend on the data and business environments.

When a mobile app runs on a user device, it is necessary to store user preferences and related configuration information in the user device itself to provide a seamless experience to the application user. It is very important to store such information securely on the user device. For Example, the API keys for the web service, sensitive private and confidential information etc. It is always better to utilize the most secured storage option provided by the device operating system.

API Keys and other sensitive information should be encrypted and stored in the



device. The data can be encrypted using a key known to the client or encoded using an encoding scheme. The level of encryption can be decided based on the business and service provided by the enterprise through the application. Encryption logic can be built, on-demand, based on the following:

- User Provided PIN
- Keys shared through a secondary channel (OTP shared as SMS/EMAIL)
- Keys securely stored in KeyStore
- Keys Stored within the Application Code (hard coded)

It is not recommended to hardcode important application URLs in the application itself. Such URLs should be supplied to the application at run time only. Hardcoded values must be stored as byte arrays and can be converted to strings of required data types at run time. It is recommended to store such hardcoded values in native layers.

The Encrypted Shared Preferences class is available in the Android Jetpack library. This library uses device-specific features for securely storing user configurations. Android KeyStore mechanism also can be used to create keys that can be used for encryption purposes. A named key can be created in KeyStore, which by default is accessible for the app which created the named key, which can be used for device-level encryption and decryption. Best practices must be followed to make the process faster as cryptographic operations are normally time-consuming. In the case of iOS applications, a keystore must be used for securely storing application-specific secrets.

App in a Rooted(Android)/ Jail Broken Device(iOS)

Jailbreaking is the process of removing software restrictions put into place by Apple on devices that run the iOS operating system. Similarly rooting in Android is the process of removing software restrictions put into place by Google and gaining the ability to access the entire operating system. A legitimate app running on a jailbroken or rooted mobile device is more vulnerable as it can expose sensitive user data. Platform-specific methods are there to detect a jailbroken or rooted phone. There is no one-size-fits-all solution for detecting jailbroken or rooted devices. There are attestation service providers who remotely evaluate the devices, whether the request is coming from the genuine app running on a genuine Android device. Attestation services may be used in the application if the business demands it.

Hiding Business Logic

Obfuscation mechanisms make it difficult to understand business logic. In software, the obfuscation of code is the process of modifying an executable so that it is no longer useful to unauthorized parties such as hackers but remains fully functional. Mobile application code, wherever possible, must be obfuscated before deployment. There are a few tools related to Android Studio such as R8, ProGuard and DexGuard. R8 is a free tool that is included in Android Studio/Visual Studio and other IDEs.

Another method is to use NDK in android and write business logic/ codes using C/C++. Decompiling and getting the source codes from NDK compiled files are very difficult.

User Awareness

Security depends on users. It is also important to educate the end-users regarding vulnerabilities that can be caused by installing apps from untrusted sources. Even legitimate apps from reliable marketplaces can include high-risk security issues. These apps can steal user information and configuration from other applications installed in the device. The security implementation in any system needs revisit and improvements regularly with increasing threats in the cyber world. Device owners must take responsibility for protecting the data they store in mobile applications. But user precautions will still fall short if developers leave vulnerabilities in their applications.

API Security

For API security, please also refer to TAG notes on "API Security".

A thorough security testing for mobile application and associated APIs must be done before moving it to production or publishing in store. Both manual and automatic testing techniques may be used. As part of automated tool-based testing, white box and black box security testing tool may be used to uncover security vulnerabilities. white box testing tool scans the source code of the application and pinpoint the line number in the application where vulnerability exists. Whereas, black box testing tool attacks the running application just like hacker and identifies security issues in the application.

References

Centre for Competence for Mobile App Development Kerala has published a document titled 'Secure API Integration for Mobile Apps' and this document is available in the digital NIC Platform of NIC.

6.0	App Testing
------------	--------------------

Mobile application testing is a procedure to test mobile applications for usability, functional and consistency glitches. Testing app on mobile devices can be done manually or with automation.

It is important for many reasons. Today's technology industry is intensely competitive, that's why it's obligatory to have and maintain an excellent quality of any kind of an application. Just a few clicks and swipes are enough for a user to praise or abandon an app. Mobile apps are important for conducting/boosting business and to make sure that mobile apps work properly, it's essential to have your applications tested in order to obtain numerous benefits.

Mobile application testing is very different from software testing and web testing. There are a few specifics to be consider before performing mobile application testing:

- **Functional Testing**
 - Business flows
 - Screen resolution
 - UI testing (e.g., landscape/portrait, languages).
 - Different device's manufacturers
 - Different OS
 - Cross-platform coverage
 - Turning on/off GPS
- **Real Environment Testing**
 - For real environment condition testing, it's very important to test on real devices instead of simulators.
 - Network conditions
 - Interruptions (e.g. calls, text messages)
 - Background/foreground
 - Gestures (e.g. force touch)
- **Non-Functional Testing**
 - Security
 - Accessibility
 - Performance and availability
 - API testing

Challenges of Mobile Application Testing

- Testing approach based on device
- Availability of multiple mobile user interfaces
- Greater security issues.
- Time constraint to market app
- Constantly changing environment and usage pattern of mobile phones
- Testing of touch screens is more complicated.

- User Experience & Issues with App Performance

There are **different approaches** you can take to these mobile testing types. This includes manual testing & automated testing, cloud testing, tool-based testing etc.

Emulators or Simulators

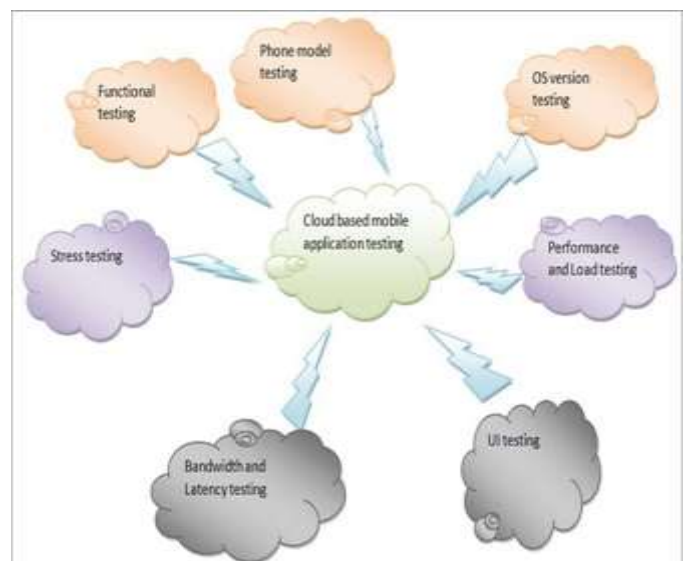
Emulators or simulators are widely used for mobile testing these days. They are tools that are capable of emulating/simulating the behaviour and working of mobile devices. An **emulator** is the original device replacement that allows you to run applications on your gadget without being able to modify them. While **simulator** doesn't imitate mobile's hardware but can set up the analogous environment as of your original mobile's OS. Hence, Simulators are preferred to test mobile application whereas Emulators are better for the mobile web application testing.

6.1 Cloud-based App Testing

Allowing the mobile application to run on several systems and networks, cloud-based testing helps to get over the drawbacks of simulators and real devices. Performance issues can also be detected by cloud-based testing.

Pros of Cloud Mobile Application Testing:

- End User gets a choice of devices, OS platforms, display densities etc. thereby reducing the overall infrastructure and maintenance cost.
- Supports parallel testing, hence saves a lot of time.
- Apps can be tested in a secure environment.
- Tools can be accessed from anywhere by both the Developer and the QA.
- Supports recording test results.
- Easy access and is available always.
- Supports testing in a real-time environment with real network carriers.



Cons of Cloud Mobile Application Testing

- Large Bandwidth
- No Backup, Redundant Tests
- Lack of Security

- Frequent Feature Changes
- Separation of Testers or Lack of Communication

Best Tools for Cloud-Based Testing

- **Kobiton:** It is an affordable, effective and highly flexible cloud-based mobile experience platform that accelerates the testing and delivery of native, web and hybrid apps on both Android and iOS. Users can execute manual and automated tests in parallel.
- **Firebase Test Lab for Android:** As the name itself suggest, it is specific to Android and can be used for any device – OS combination of Android. It comes in both free and paid versions. The test results can be in the form of logs, video, and screenshots. It is a suitable tool for Continuous Integration (CI). Although Firebase Test Lab is also available for iOS as separate service.
- **AWS Device Farm:** This is a leading cloud-based testing tool made by Amazon and can be used for both Android and iOS. This can be used for testing web, native and hybrid type of apps. The reports are generated in the form of video, logs, screenshots etc. and can be run on real and non-rooted phones. It is a paid tool.
- **Perfecto:** It provides manual, automation and performance testing on real devices. It doesn't support emulators. It is a good choice for both Android and iOS and it can support more than a thousand real devices. It provides a plug-in for Jenkins, Eclipse, and Appium. It is a paid tool.

AWS and Firebase Test Lab services keep on maintaining the logs which are always available publicly. This aspect is worth considering while opting for these services.

6.2	Stages of App Testing
------------	------------------------------

1. Documentation Testing

The beginning of mobile testing takes place from Documentation testing - preparatory stage.

Even before the development of the app starts the testers are handed over screen layouts, navigational charts and other requirements that are obscure on the design.

In this phase, you need to analyse the requirements for wholeness and discrepancy. All the discrepancies found in this stage are required to be resolved before the development begins.

Documentation phase marks the creations and analysis of requirements (Specification, PRD), Test Cases, Test Plan, Traceability Matrix.

2. Functional testing

It helps you test whether your mobile application works as expected and in accordance to the requirement specifications. While you are performing functional testing for your app keeps the following factors in mind:

- Business functionality of your app like banking, social networks, education etc.
- Target audience like companies, students, entrepreneurs, etc.
- Distribution channels like Google Play, direct delivery, App Store, etc.

The **basic validations** that you need to test in functional testing are:

- Installing and running the application
- Fields testing
- Business functionalities testing
- Interruptions testing
- Constant users feedback testing
- Update testing
- Device resources testing

3. Usability Testing

Usability testing ensures that your application offers convenient browsing to your customers and creates an intuitive interface that abides by industry standards. It promises fast and easy-to-use applications. Usability of your application is judged or based on three basic criteria:

- Satisfaction
- Efficiency
- Effectiveness

4. UI (User Interface) Testing

User Interface (UI) testing ensures that your application's GUI meets all the required specifications.

5. Compatibility (Configuration) Testing

Compatibility (Configuration) testing validates the optimal performance of your application on different devices based on their size, screen resolution, version, hardware, etc. Compatibility testing also takes care of

- OS Configuration
- Browser Configuration
- Database Configuration
- Device Configuration
- Network Configuration

Compatibility Testing can be further divided into

- **Cross-platform Testing:** Testing your mobile application compatibility with different operating systems like Windows, iOS, Android and BlackBerry etc.
- **Cross-browser Testing:** Testing your mobile application compatibility in different browsers like Google Chrome, Mozilla Firefox, Opera etc.
- **Database Testing:** Testing your mobile application compatibility in different database configurations like DB2, Oracle, MSSQL Server, MySQL, Sybase.
- **Device Configuration Testing:** Testing your mobile application compatibility on different devices based on
 - **Device type:** Smartphone, tablet, etc.

- **Device configuration:** Processor, RAM, battery capacity, screen resolution, etc.
- **Network configuration Testing:** Testing your mobile application compatibility in different network configurations (TDMA, GSM, CDMA) and standards (2G, 3G, 4G, 5G).

6. Performance Testing

Performance testing helps you test your application reaction and constancy under the specific workload.

Different Attributes of Performance testing are as follows:

- **Load Testing:** It is done to check the application's behavior under normal and extreme loads.
- **Stress Testing:** It is done to assess the performance of an app in case the CPU usage, system memory, or other hardware specs are reaching their limits. The goal of such tests is to validate the app's behaviour at its peak condition.
- **Stability Testing:** It tests if your application can work well for a longer period within normal loads.
- **Volume Testing:** It is conducted to test your application's performance when subjected to a huge volume of data.
- **Concurrency Testing:** It tests the performance of your application when multiple users are logged in.

7. Security Testing

Security testing validates the security features of your application. It also analyses the risks of application hackers, protection, viruses and unauthorized access to extremely sensitive data.

8. Recovery Testing

Recovery testing tests the ability of your application to withstand and successfully recover from possible and potential failures caused by software issues, hardware failures or communication problems.

9. Localization Testing

Localization testing tests the adaptability of your application for a specific target audience based on cultural specifics.

10. Change related Testing

Finally, when complete testing is done, you might find some bugs, resulting in a certain piece of code to change to eliminate those bugs. After these code changes you again need to carry out a round of testing. This basically includes:

- **Re-testing or Confirmation Testing:** To test that all the detected defects are successfully fixed.
- **Regression Testing:** Sometimes code changes can even disturb the working of existing and properly working functions. Regression testing is done to ensure that the new changes did not lead to the appearance of new bugs.

11. Beta Testing

Beta testing is done by real users on real devices to validate usability, functionality, compatibility and reliability testing. Before pushing your app forward for beta testing, take account of the following factors

- A number of testing participants
- Testing duration
- Shipping
- Demographic coverage
- Testing costs

Beta testing is good investment ensuring a better quality of your mobile app.

12. Certification Testing

Certification testing tests whether your application meets the standards, licensing agreements, terms of use and requirements of stores like the App Store, Google Play and Windows Phone.

Manual or Automated Testing

Some testers till date support manual testing while other believe it is going to die. But the fact is that both automation testing and manual testing are important. There are scenarios that are best tested with automation testing, but there are few that give the finest results with manual testing.

Manual Testing: Manual testing is the process in which QA analysts execute tests one-by-one in an individual manner.

Automation Testing: Automation testing is the process in which testers utilize tools and scripts to automate testing efforts. Automation testing helps testers execute more test cases and improve test coverage.

Automation vs. Manual Testing

- If the application contains new functionality, test it manually.
- If the application requires testing once or twice, do it manually.
- Automate the scripts for regression test cases. If regression tests are repeated, automated testing is perfect for that.
- Automate the scripts for complex scenarios which are time-consuming if executed manually.
- Manual testing is time consuming and less efficient as everything is to be done manually and it will be difficult to ensure sufficient test coverage.
- Automated testing can be done using tools and scripts and is more efficient as it covers real user simulations for greater test coverage.

Manual Mobile Testing

Manual testing is irreplaceable when it comes to functionality and first impressions. It gives you the kind of feedback you might hear from the actual users. The most significant merits of manual testing are flexibility and

simulation of real user actions. But when is manual mobile testing applicable, and when it is better to choose test automation? Here is a list of testing types that resonate with manual mobile testing.

- **Usability and UI Testing.** Only humans can evaluate how user-friendly and handy the app is. Even a slight divergence between the actual look and feel of the app and the design agreed upon in the specs can be noted by an experienced manual tester right away. A machine can't do that.
- **Compatibility Testing.** This type of mobile testing can be automated. However, it is usually not a repetitive task; thus, doing it manually makes sense because manual app testing is more flexible and cost-effective. It's also important to check if the app is responsive on real devices with different OS versions, screen sizes and resolutions.
- **One-time Testing.** Writing a script is not at all profitable if you need to test some features only once or twice. If the task is not repetitive or monotonous, it's better to run the test manually and document the results.
- **Localization (L10N) and Internationalization (I18N) Testing.** Since the font or word length varies from language to language, it may affect the UX/UI of the app (especially when it comes to UI mirroring). Manual testing proves useful when you need to check if the app performs well in all languages and regions that are in the requirements.
- **Ad-hoc Testing.** Both of these types of testing require experience, intuition and creativity of testers since there is no script or scenario that they can use. A professional quality assurance engineer can foresee the possible actions an end-user will make and ensure the app runs smoothly in these situations.

On the flip side, manual mobile testing is not the best option for:

- Regression testing (whereas automated testing works perfectly for that)
- Load and performance testing
- Complex scenarios with large data sets that are resource- and time-consuming if executed manually
- Functional testing for large applications that need to be run frequently
- Being the main testing approach to achieving continuous testing

Steps to Test Mobile Apps Manually

Manual testing is sometimes deemed as randomly clicking or tapping through the mobile app and logging bugs. Well, if you don't have a plan or structure, then yes, your manual mobile testing will consist of mere clicking around. So, to keep everything under control, you need to outline your action plan and define the route for executing it.

Here are the five key steps you need to take when testing an app manually.

1. Have a Plan

It may sound obvious, but you need to know exactly what to test and how to test it beforehand. Otherwise, it will be like running in circles trying to catch any bug. So, start with studying your test plan and choosing the test cases that are better to test manually. Then, group them based on what should be tested (navigation, UI elements, interruptions, etc.). This will help you keep your test sessions as brief as possible.

2. Outline a Click Path

For each use case, write down a detailed click path. It will let you optimize the time per each test session when repeating precisely the same test on different devices and in case you discover a bug, you'll be able to reproduce it easily and quickly.

3. Test on a Variety of Devices

An app can work flawlessly on one device but crash on another. That's why it is crucial to perform the same test on different real devices with different OS, hardware and software options. Besides, in the case of Android OS, some manufacturers make custom UI changes that may affect the app (for example, Xiaomi MIUI or Huawei EMUI). You can use real devices in the cloud and choose the parameters you need.

4. Repeat

Repeat all the steps in the same sequence again in case new bugs arise after code modification. It is important to run all the tests to be sure that fixing one issue didn't break anything else.

5. Document the Results

Make sure to write down all the information you have gained so that you can easily test that part again after the bug has been fixed.

Mobile App Testing Process



- **Requirement Analysis:** Analyze your project requirement to identify functional features and type of the mobile app (Native, Web, or Hybrid).
- **Test Strategy & Planning:** To ensure reasonable and vast machine coverage, we select the target mobile devices & users based on the business requirements.

- **Designing Test Cases & Scripts:** We draft test suites and test cases for various features and functionalities.
- **Testing Parameters:** Before testing any mobile app, we freeze upon features, functionality and more with related attributes, parameters & reporting mechanism (PM & Development teams) to report bugs.
- **Analyzing Test Results:** We provide a summary on test report & analysis across scenarios throughout the entire development lifecycle.

Choosing the Right Mobile Test Automation Framework

A critical decision for organizations that are defining their automated testing strategy is choosing the right automated testing framework that will help their development and QA teams efficiently write and execute automated tests. The market offers a plethora of frameworks to choose from, and the decision isn't always easy.

Criteria for Selecting a Mobile Test Automation Framework

As you begin evaluating mobile testing frameworks to determine what's suitable for your organization, here are some key questions you should ask to help uncover your needs and understand the impact of a new tool on your organizational processes, tech stack and teams:

- **Organizational process:** What does the process for buying/vetting a tool in your organization entail? Is there a separate team that can guide your research process and help, or do you have the freedom to choose?
- **Future needs:** Will the new framework address your evolving business and testing needs in the future.
- **Application type:** What is the type of application you are developing and need to test: native, hybrid, web?
- **Requirements:** Does the framework meet your usability, speed, parallel testing, and hosting requirements?
- **Integration:** Does the framework integrate well in your CI/CD pipeline?
- **Environment and tech stack:** How do existing tech stack look? Does the new framework support the technology and environment you are using?
- **Teams:** The structure of your team and how it functions are important considerations when choosing a new tool. Does the team consist of?
 - Mainly developers embracing the shift-left motion?
 - Developers and technical QA teams following agile development?
 - Developers and manual QA currently in a transition to agile?
 - Siloed QA and dev teams following the more traditional waterfall approach?
- **Testing responsibilities:** Who writes the tests? What is the testing process followed by your team? Will the framework support these specific needs?

- **Resource skill sets:** Does your team have the required skill sets to work with the testing framework? Will they require special training and onboarding? How will the learning curve look?
- **Budget:** What are the costs you need to consider with each framework?
- **Getting started:** Can you evaluate the testing framework with a free trial? How complex is the framework's workflow?
- **Community and support:** Do the vendor provide good online support and resources? Does it have an active community to help troubleshoot your issues?

The **common mistakes** that teams tend to make are:

- Getting a framework and then hoping it will work for your team.
- Choosing it because you have used it before and it's familiar.
- Going after a popular brand (If x built it, it must be good).
- Selecting one because it is currently hot in the market.
- Setting out to do it all on your own and building a new framework.

Popular Automation Mobile App Testing Tools

The most popular Mobile App Testing Tools are:

- For **Functionality Testing:** Appium, Selendroid, Robotium, Ranorex.
- For **Usability Testing:** Reflector, User Zoom, Loop.
- For **Interface Testing:** iMacros, FitNesse, Jubula, Coded UI, LoadUI.
- For **Compatibility Testing:** CrossBrowserTesting, BrowserStack, Browsera, Litmus, Rational ClearCase, Ghostlab.
- For **Performance Testing:** Apteligent, NeoLoad, New Relic.
- For **Security Testing:** OWASP Zed Attack Proxy, Retina CS Community, Google Nogotofail, Veracode, and SQL Map.

6.3	Android Testing Frameworks
------------	-----------------------------------

There are a handful of Android Testing Frameworks that are available. Some of the most commonly used frameworks are:

1. Robotium Test Framework

- This framework is used to write sophisticated and robust black box test cases for Android applications.
- It supports both native as well as hybrid clients.
- Functions, System test cases, and User Acceptance test cases can be written using this framework.
- Robotium supports Android 1.6 and above and also support for Dialogs, Menus, Activities etc.
- This framework handles multiple Android activities automatically.

- A handful of methods are given as a part of Robotium for interacting with different graphical components of the Android application. Some of them are:
 - goBack();
 - getButton();
 - isRadioButtonChecked();
 - searchText("User");
 - click on button("Logout");

2. Appium Test Framework

- This framework works for native, hybrid and mobile-web apps for Android devices.
- Appium is free to use utility.
- Single API works for both Android as well as the iOS platform. This is one of the frameworks which **supports cross-platform** testing.
- It uses Selenium Web driver to interact with the Android application.
- Appium supports script writing using a lot of programming languages such as Java, C#, Python, PHP, Ruby etc.

3. Espresso

- Espresso is the testing framework that comes built into Android Studio and is designed specifically for functional testing of Android applications.
- It is easy to set up, more stable than Appium and gives them the ability to quickly test code components.
- With its automatic synchronization of UI, Espresso allows for faster test execution.
- It brings faster feedback for developers, as it does not require a server for communication.
- Espresso is only compatible with Java and Kotlin, meaning that your tests can only be written in these languages.
- If your team is developing an app that will be listed on both iOS and Android, you will need to find another framework to help ensure compatibility across these different operating systems.

6.4	iOS Testing Frameworks
------------	-------------------------------

Some of the most commonly used iOS Automation Frameworks are listed below:

1. Appium

- Appium uses Selenium Web driver to automate iOS application testing.
- This platform is independent and can be used both on the web and mobile devices (both Android and iOS).
- This is an Open Source one and is not restricted by language.

- Application changes or source code access is not required for automation using Appium.
- Appium works seamlessly independent of the application type: be it, Native, Hybrid or Web.

2. XCUITest

- XCUITest is the testing framework that ships with Apple's XCode development system and is the most popular framework for developers who want to test their iOS apps.
- It offers a comprehensive functionality to do basic unit level testing of iOS apps.
- Due to the framework's architecture, test execution against iOS devices is faster, with less flaky tests and more reliable results.
- Simplifies the testing process by allowing teams to work on the app's source code and test code in one place.
- It is simpler to work with, intuitive to set up and easy to maintain when compared to a framework like Appium.
- It is dedicated to Objective-C and Swift, which normally runs exclusively under iOS.
- It gives developers an easy method to run tests on the quality of their code, QA tends to move away from the iOS lock-in of XCUITest and instead opts for more OS-agnostic testing frameworks.
- The scope of testing with XCUITest is also limited to the application, while with Appium, users can test the complete flow and interaction between apps, browsers, and OS.

3. Calabash:

- Calabash is an Open source cross-platform framework which supports both Android and iOS automation testing.
- Calabash tests are written in Cucumber which is similar to that of a specification and is easy to understand.
- Calabash consists of libraries which enable the users to interact with both native and hybrid applications.
- It supports interactions such as gestures, assertions, screenshot etc.

4. Earl Grey:

- Earl Grey is Google's own internal UI testing framework.
- This has been used for testing YouTube, Google Photos, Google Play Music, Google Calendar etc.
- The major advantages of Earl Grey are, Build-in synchronization, Visibility checks before interactions, true user interaction (Tapping, swiping etc.).
- This is very similar to Espresso by Google which is used for Android UI automation.

5. UI Automation:

- UI Automation is developed by Apple and is very similar to UI Automator to Android.
- The APIs are defined by Apple and the tests are written in JAVA.

6. KIF:

- KIF stands for "Keep It Functional" is a third party and Open source framework.
- This is an iOS integration test framework which is closely related to and used for XCTest test targets.
- KIF is easy to configure or integrate with the Xcode project and thus additional web server or additional packages are not required.
- KIF has a wide coverage in terms of iOS versions.

TestFlight Beta Testing Overview (iOS, macOS)

TestFlight Beta Testing lets you distribute beta builds of your app to testers and collect feedback. You can enable TestFlight beta testing for up to 100 apps at one time in your App Store Connect account. Make improvements to your app and continue distributing builds until all issues are resolved before you submit your app to the App Store.

Note: Your macOS apps must be built with Xcode 13 or later to use TestFlight for Mac.

Step 1: Enter your Test Information

Enter test information about your app, such as a description and feedback email. You will need this if you plan to distribute your build to external testers (persons outside your team).

Step 2: Upload your Build

Upload your build to the App Store. Read Cryptography and U.S. Export Compliance to determine if you need to provide export compliance documentation for your app. After builds are uploaded, they are available for testing for 90 days.

Step 3: Invite Internal and External Testers

You can create groups for internal and external testers, then assign specific builds to them. After you've added builds to a group, you can add external testers (up to 10,000 people) and internal testers (up to 100 App Store Connect users with access to your content) to test your app. If you invite external testers, the build needs to be approved by TestFlight App Review before testing can begin.

Step 4: Testers download TestFlight and Accept your Invitations

Testers install the free TestFlight app on their devices. Then they use their invitation email or a public link to accept invitations, install your app, send

feedback, and get updates. Testers download and install thinned variants of your app.

Step 5: View Tester and Build Information

Track your tester engagement and your app's performance by viewing build status and metrics in App Store Connect—such as numbers of sessions and crashes. You can also resend email invitations to testers who have not yet accepted their invitation.

Step 6: Collect Feedback from Testers

Testers running TestFlight for iOS, version 2.3 or later and iOS 13 or iPadOS 13 or later, can send feedback through the TestFlight app or directly from your beta app by taking a screenshot. This is also available in TestFlight for Mac (Mac with M1 Chips). You can view this Tester Feedback in the Feedback section in App Store Connect.

Enter Test Information for External Testing

If you distribute your app to external testers, you need to enter additional TestFlight test information about your app for TestFlight App Review. You can enter this information when you add your app to your account or before you invite external testers.

Required Role: Account Holder, Admin, App Manager, Developer, or Marketing.

- From **My Apps**, select your app.
- Click the **TestFlight** tab.
- In the **sidebar**, under **General Information**, click **Test Information**.
- On the right, choose a language option and enter required test information.

The information you enter for your beta app can be different from the information you enter later when submitting your app to the App Store. In the Beta App Description text field, enter a description of your beta version. In the Feedback Email field, enter the email address that testers can use to contact you through the TestFlight app. This is also the reply-to address in email invitations to testers.

7.0	Publishing Your App
------------	----------------------------

Once you develop the mobile app, the next step is to make it available to the end-users. The App will normally be used by the citizens, businesses, employees and Government. Presently, Apps are made available to users through Google Play Store or Apple App Store. Link to the App(s) is/are given on the relevant Government Website(s). The NIC developed Apps are linked through the **NIC Mobile App Store** at <https://egovmobileapps.nic.in> website.

It is mandatory to host NIC developed Apps only in the NIC accounts available on the google play store and apple app store. Access to NIC Google Play Store account is given to all Centres of Competence for Mobile App Development while Apple Apps are hosted through Kerala Centre. Therefore, you need to get in touch with the respective Centres assigned to your State for hosting of the Mobile App. The procedure to host these Apps through NIC account is given below, which gives a fair idea of the content that should be submitted along with the AAB/APK to the Competency Centre.

7.1	Publishing on Google Play Store
------------	--

First, Create New App as following:

- Sign into Google Play Console Developer Account.
- The list of all apps created (under the account) will be shown.
- Tap on 'Create App' to create the new app and fill all the details.
- **App Details**-Fill the following details:
 - **App Name:** This is title/name with which your app will appear on Google Play. It should be concise and not include price, rank, any emoji or repetitive symbols. The maximum number of characters for app name is now 30 which were 50 Characters till now. Although the Google Play Store restricts the uniqueness of package name assigned to the app being hosted but it is always recommended to check the existence of mobile app with the same/similar name you have chosen for the app to use unique name as well.
 - **Default Language:** Choose the default language of your app.
 - **App or Game:** Select whether your app is an app or game.
 - **App Pricing:** Select whether the app is available for free or it is paid.
 - **Accept the Developer Program Policy** User need to provide the self-declaration on:
 - This application meets Android Content Guidelines. Please check out these tips on how to create policy compliant app descriptions to avoid some common reasons for app suspension. If your app or store listing is eligible for advance notice to the Google Play App Review team, contact us prior to publishing

- US Exports law
 - I acknowledge that my software application may be subject to United States export laws, regardless of my location or nationality. I agree that I have complied with all such laws, including any requirements for software with encryption functions. I hereby certify that my application is authorized for export from the United States under these laws.

The app will be created and dashboard of the App will be shown.

- **Dashboard:**

- The dashboard shows the different steps of app to be completed for the app to be hosted.
- The dashboard also shows app information like app performance, store listing performance, rating and reviews etc.

- **Main Store Listing:**

Edit your App's name, icon, screenshots and more to present how your app looks to users on Google Play. All the fields are to be entered in English language only. Feature graphic, screenshots, short description, and videos are used to highlight and promote your app on Google Play and other Google promotional channels.

- App Details
 - App Name: Already discussed above.
 - Short description: A short description of your app. Maximum of 80 Characters.
 - Full description: The full description of your app. Maximum of 4000 Characters.
- Graphics
 - App icon: This is the App icon which will be shown on Google Play. Don't use badges or text that suggest store ranking, price or Google Play categories (such as "top", "new", or "sale"). The app icon should be
 - Transparent PNG or JPEG
 - 512 px by 512 px
 - Up to 1 MB
 - Feature graphic: This will help promote your app in different places on Google Play. In case of cropping, avoid placing text near the edges, and centre key visuals. The Feature graphic should be
 - PNG or JPEG
 - 1024 px by 500 px
 - Up to 1 MB
 - Video: The YouTube link for your App video can be provided. It is not mandatory. The video should be short and may contain the demo of App, functionality etc.

- **Screenshots:** The Screenshots for your App are needed to showcase your App's features and functionality on its store listing page, which can help your App to attract new users on Google Play. Screen shots of different screen sizes can be added as given below:
 - **Phone:** The following guidelines need to be followed for uploading up to 2–8 phone screenshots. The screenshots must be
 - PNG or JPEG File
 - 16:9 or 9:16 aspect ratio
 - Each side between 320 px and 3,840 px
 - Up to 8 MB per screenshot
 - **Tablet:** The screenshots for tablet are Non-Mandatory. If you are uploading the screenshots for the tablet, then the following guidelines needs to be followed. The screenshots must be
 - PNG or JPEG Files
 - Can Upload up to 8, 7-inch tablet screenshots
 - 16:9 or 9:16 aspect ratio
 - Each side between 320 px and 3,840 px
 - Up to 8 MB per screenshot
- **Privacy Policy**
 - The Privacy policy helps provide transparency about how you treat sensitive user and device data. The privacy policy is required by Google if any of the below permissions are being used within the App:
 - Camera
 - Microphone
 - Contacts, Accounts
 - Calendar
 - Location
 - Sensors
 - Storage
 - Messaging
 - Phone State

Please note that Google Keeps on changing the policies and may add/remove such requirements for any permission any time for new apps and gives time range to adhere to the new policy guidelines for the existing apps.

- **Ads**
 - Whether the app contains any advertisements or not.

- The ads can be visible on Google Play store, next to the app, under the 'Contains ads' label. Such apps need to adhere to the advertising policy guidelines as released by Google Play Store.
- **App Access:** Provide instructions on how to access restricted parts of your app if any or all the functionality of the app is not available without special access.
 - If parts of your app are restricted based on login credentials, memberships, location, or other forms of authentication, provide instructions on how to access them. The app developer can provide up to 5 such combinations covering roles. Make sure this information is kept up to date. The developer need to create sample user credentials for the purpose of app review carried out by Google Play Store and is mandatory.
 - Provide the following details for App Access
 - Name
 - Username/Mobile Number
 - Password
 - Any Other Instruction
 - Google may use this information to review your app. It won't be shared, or used for any other reason.
- **Content Rating:** Your App's ratings are displayed on Google Play to help users identify whether your App is suitable for them. The questionnaire is to be filled to evaluate the rating for the app as per following criteria:
 - Violence
 - Does the app contain violent Material?
 - Can this violent material be visually depicted?
 - Can this violent material be referred to through text or spoken about?
 - Sexuality
 - Does the app contain sexual material or nudity (Except in a natural or scientific setting)?
 - Can this sexual material be visually depicted?
 - Can this sexual material be referred to through text or spoken about?
 - Language
 - Does the app contain any potentially offensive language?
 - Is this content the focus of the app?
 - Can the app contain mild and/or language that could be considered offensive?
 - Can the app contain discriminatory language?

- Can the app contain sexual expletives?
- **Controlled Substance**
 - Does the app contain references to or depictions of illegal drugs?
 - Is this content the focus of the app?
 - Can this app contain visual depictions of illegal drugs?
 - Can illegal drugs be referred to through text or spoken about?
- **Miscellaneous**
 - Does the app natively allow user to interact or exchange content with other users through voice communications, text or sharing images or audio?
 - Does the app share user-provided personal information with third parties?
 - Does the app share the user's current physical location to other users?
 - Does the app allow user to purchase digital goods?
 - Does the app contain any Nazi symbols, references, or propaganda?
 - Is the app a web browser or search engine?
- **Target Audience and Content**
 - This helps make sure that apps designed for children are safe and appropriate.
 - Select the respective age group for which the app is made.
 - The privacy policy is required for the age group below or equal to 15 years.
 - Appeal to Children
 - Whether your app appeals to children or not.
 - 'Not designed for children' warning appearing on some Play Store listings.
- **News App**
 - Whether your app is a news app? Yes/No
- **COVID-19 Contact Tracing and Status Apps**
 - Whether the app is a COVID-19 **contact tracing** or **status app**? Yes/No
- **Data Safety**

Google Play Store has recently (November 2021) introduced the Data Safety questionnaire where the type of data being collected in the app and its usage in the app needs to be declared as per new policy.

 - Data Collection and Security

- Does Your App Collect or Share any of the required data types? Yes/No
- Is all of the user data collected by your app encrypted in transit? Yes/No
 - Do You Provide a way for users to request that their data is deleted? Yes/No
- Data Types

Select the appropriate data type being Collected or Shared by the app

 - ✓ Location (Approx. location, precise location)
 - ✓ Personal Info (Name, Email, Phone No, Address, Personal Identifiers, Race & Ethnicity, Political or Religious beliefs, Sexual Orientation or gender Identity, other personal info)
 - ✓ Financial info (Purchase History, credit info, credit/debit card or bank account number)
 - ✓ Health and Fitness (Health information and personal information)
 - ✓ Messages (SMS or MMS messages, Emails, other In-App messages)
 - ✓ Photos and Videos
 - ✓ Audio files (voice recordings, music files, other audio files)
 - ✓ Files and docs
 - ✓ Calendar (Calendar events)
 - ✓ Contacts
 - ✓ App activity (Page views and taps in app, Installed Apps, in app search history, other user generated content)
 - ✓ Web Browsing (Web Browsing History)
 - ✓ App info and performance (Crash Logs, diagnostics, other app performance data)
 - ✓ Device or Other identifiers (Model, IMEI no. MAC Address etc.)
- Data Usage and Handling

The consent form for each of the selected data types need to be filled individually. The following information needs to be filled in consent form:

 - Is Data Collected, shared or both.
 - Is data processed ephemerally? Yes/No
(means data is stored only in memory, and is retained no longer than necessary to service the specific request in real time.)
 - Is this Data required for your app, or user can choose?
 - ✓ Data Collection is required (users can't turn off data collection)
 - ✓ Users can choose whether this data is collected
- Why is this user data collected (Select all that apply)?
 - ✓ App Functionality (used for features in your app)

- ✓ Analytics (used to collect data how users use your app or how your app is performing)
- ✓ Developer Communication (used to send news or notification about your app)
- ✓ Fraud Prevention, security and compliance (used for fraud prevention, security or compliance with laws)
- ✓ Advertising or marketing (used to display or target ads or market communication)
- ✓ Personalization (used to customize, such as showing recommended content)
- ✓ Account Management (used for setup and management of users account)
- Store Listing and Preview

This page shows the preview of all the information filled under data safety. All that information will be shown to users on Google Play.
- Privacy Policy

Also, the privacy link clearly specifying the collection or sharing of data to be mentioned. Please make sure your privacy policy is available on an active URL (no PDFs) and is non-editable.
- Country/Region
 - Select the country(s) for which your app will be available.
- Store Listing Contact details
 - Provide the Contact details regarding your app. The details will be published on Google Play and can be viewed under Additional Details of the app, so that users can contact you for any type of queries.
 - Email Address
 - Phone Number (optional)
 - Website URL (optional)
- Authority Letter: Google Play wants to help you ensure that if your app communicates government information, it does so transparently and empowers users with reliable and clearly sourced civic information. Google doesn't allow apps that contain false or misleading information or claims, including in the app description, title, icon, and screenshots. This includes apps that falsely claim affiliation with a government entity or offer to provide or facilitate government services for which they are not properly authorized.

So to protect users from any false information, authority letter from the respective Government or department for which app is made, needs to be submitted to Google.

- Production: Create New Release

- Upload the AAB File (*Google do not allow APK file for any new app w.e.f. 1st August 2021*)
- The app must have "target SDK version" 30 and "minimum SDK version" 17. (*Google revise these criteria based on the latest security policy updates from time to time*)

To finally submitting the new app for Google Play Store Review and making it available on the Play Store, Review Release has to be tapped and provide short text related to the app, which will be visible to the user under store listing, and submit the app. The Play Store team reviews the app and on successful completion of the review, app is made available for downloads on Google Play Store. The play store team sends objections, if any, to the developer's gmail account. Such objections are also made available under the developer console against such app. The developer has to work on the raised objections and re-submit the revised AAB after removing the reported problems in the app.

7.2	Recent Changes in Store Policies
------------	---

Google Play changes its policies from time to time. Currently Google has introduced the following new policies that needs to be declared within the app content.

News App

Whether your app is a news app? This helps in ensuring the transparency in news publishing on Google Play, and makes the difference between news apps and other types of apps clearer.

COVID-19 Contact Tracing and Status Apps

Whether the app is a COVID-19 contact tracing or status app? The apps that track or monitor infected or exposed individuals, and apps that verify an individual's current infection or vaccination status.

Data Safety

This helps in showing app's privacy, security and data handling practices before the user download it. The data being collected or shared, uses different **data types** like name, email, location, photo, video, SMS etc., wherever available, for an associated runtime permission.

The separate consent for each of the declared data types must be submitted, clearly stating how the data collected from app is being stored or shared and what's the purpose of collecting or sharing the data; whether it is required for advertising or it's the part of main app functionality etc.

The in-app disclosure of data access, collection, use, and sharing must be provided. The app's request for consent:

- Must present the consent dialog clearly and unambiguously;
- Must require affirmative user action (e.g., tap to accept, tick a checkbox);

- Must not interpret navigation away from the disclosure (including tapping away or pressing the back or home button) as consent; and
- Must not use auto-dismissing or expiring messages as a means of obtaining user consent.

This information is shown on store listing to help users understand how the app collect and share their data.

Privacy Policy: The privacy policy link needs to be provided comprehensively disclosing how your app accesses, collects, uses, and shares user data with any of the third parties with any personal or sensitive user data.

Apps that do not access any personal and sensitive user data must still submit a privacy policy.

Please make sure your privacy policy is available on an active URL (no PDFs) and is non-editable.

7.3	Publishing on Apple App Store
------------	--------------------------------------

First, Create New App Against Bundle ID/Package name as following:


- Sign into Apple Developer Account (<https://developer.apple.com/account>).
- Tap on 'Certificates, Identifiers & Profiles'.
- Tap on 'Identifiers' to create register App Bundle ID on Apple Store Listing.
- Once Bundle ID is registered as Identifier, then **Generate a Provisioning Profile** under 'Profiles' -> 'Distribution' -> 'App Store' and select Registered Identifier under 'Select an App ID' Drop Down.
- Download and open the Profile generated with Xcode onto your registered Mac.
- Now sign into App Store Connect (<https://appstoreconnect.apple.com/>).
- Under 'My Apps' Click '+' icon and select 'New App'.
- **'New App'**-Fill the following details:
 - **App Name:** This is title/name with which your app will appear on App Store(iTunes). It should be concise and not include price, rank, any emoji or repetitive symbols. The maximum number of characters for app name is 50 Characters.
 - **Primary Language:** Choose the Primary language of your app.
- **App Details**-Fill the following details:
 - **Platforms:** Select at-least on platform
 - **iOS:** iPhone, iPad, Apple's Watch or M1 based Mac Books/iMac (with MacOS version 12.0 or above)
 - **macOS:** MacBook, iMac, Mac-Mini.
 - **tvOS:** Apple TV.
 - **Name:** This is title/name with which your app will appear on App Store(iTunes). It should be concise and not include price, rank, any

emoji or repetitive symbols. The maximum number of characters for app name is 50 Characters.

- **Primary Language:** Choose the Primary language of your app.
- **Bundle ID:** Choose your registered Identifier here.
- **SKU:** A unique name for your app (not visible to anyone).
- **User Access:**
 - **Limited Access:** Add apple ID to whom you want to make your app visible on App Store.
 - **Full Access:** All will be available to all apple users into their App Store.
- Click on `Create` button and app will be listed in your app store connect account.
- Open your iOS app project into Xcode and select `Any iOS Device` as simulator.
- From Menu bar go to Projects -> Archive.
- Once app is archived a new window named `Organizer` will appear.
- Click `Distribute App` button and Select a distribution method (Select App Store Connect) and Upload the archive.
- Once Upload finishes, again go to app store connect and start filling other details about app as below:
- **App Details**-Fill the following details:
 - **App Previews and Screenshots:** Add App screen shots as per devices as attached.



- **Promotional Text:** Promotional text lets you inform your App Store visitors of any current app features without requiring an updated submission. This text will appear above your description on the App Store for customers with devices running iOS 11 or later, and macOS 10.13 or later.
- **Keywords:** Include one or more keywords that describe your app. Keywords make App Store search results more accurate. Separate keywords with an English comma, Chinese comma, or a mix of both.
- **Description:** A description of your app, detailing features and functionality.
- **Support URL:** A URL with support information for your app. This URL will be visible on the App Store
- **Marketing URL:** A URL with marketing information about your app. This URL will be visible on the App Store.

- **Build:** Add App Build by clicking  Icon.
- **General App Information:**
 - **Version:** The version number of the app you are adding. Numbering should follow software versioning conventions.
 - **Copyright:** The name of the person or entity that owns the exclusive rights to your app, preceded by the year the rights were obtained (for example, "2021 National Informatics Centre"). Do not provide a URL.
 - **Game Center:** If Applicable.

App Review Information:

Provide a user name and password so we can sign in to your app. We'll need this to complete your app review.

- **Contact Information:** The person in your organization who should be contacted if the App Review team has any questions or needs additional information.
 - **Sign-in required:** Please Check only if app requires login testing. And provide a test user credentials.
 - **Notes:** Additional information about your app that can help during the review process. Include information that may be needed to test your app, such as app-specific settings.
 - **Attachment:** You can attach specific app documentation, demo videos, and other items to help prevent delays during the app review process. Make sure you use files with the following extensions: .pdf, .doc, .docx, .rtf, .pages, .xls, .xlsx, .numbers, .zip, .rar, .plist, .crash, .jpg, .png, .mp4, or .avi.
 - **Version Release:** This app version can be automatically released right after it has been approved by App Review (Default). You can also manually release it at a later date on the App Store Connect website or in App Store Connect for iOS.



- **Pricing and Availability**
 - The price determines the App Store price and your proceeds. If your app is free, choose Free. If you sell your app, you must have a Paid Application agreement.:
- **App Privacy**
 - **Privacy Policy URL:** A URL that links to your privacy policy. A privacy policy is required for all apps.
 - **Data Collection:** Do you or your third-party partners collect data from this app?

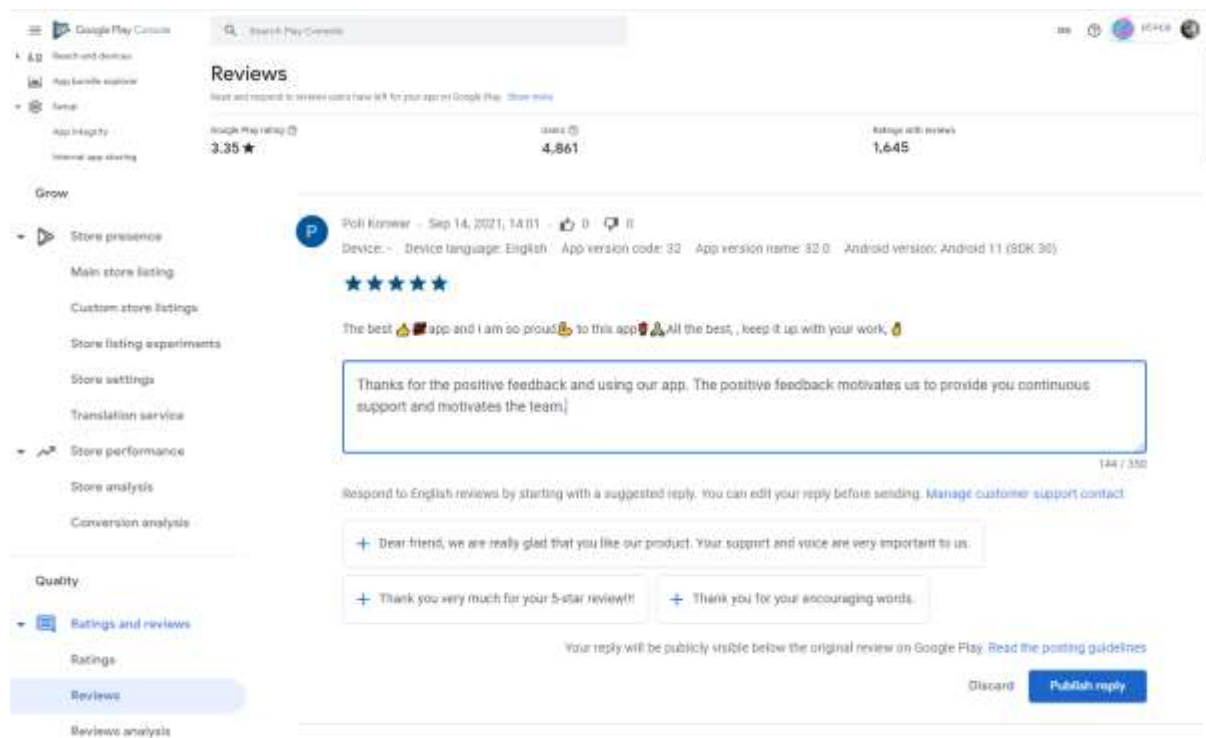
- **Yes**, we collect data from this app.
- **No**, we do not collect data from this app.

To finally submitting the new app for Apple Store Review and making it available on the Apple Store Click `**Submit For Review**` button.

The Apple team reviews the app and on successful completion of the review, app is made available for downloads on Apple Store. The Apple team may send objections, if any, to the developer's email account or can also accessible on App store connect Resolution Centre. The developer has to work on the raised objections and if Objection is of type `**Binary Rejected**` developer required to re-submit the revised app after removing the reported problems in the app else for any other objections developer have to just reply in the same **Resolution Centre**.

8.0 Maintenance, Reviews, Enhancements

Once any app is available on the Play Store, the developer must keep a close look on the User Reviews and Ratings app is receiving on the play store from app users. Another important aspect to keep the app visible on app store is to keep your app up to date to adhere to latest Google Policies amended from time to time which may be related to latest security updates, release of new version of libraries/APIs.



Whenever, the app requires modifications to meet the user expectations or to adhere to latest Google Policy guidelines or may be due to introduction of new features in the app or may be change in UI/UX, a new version of the app needs to be released. Mechanism of updating App, whenever new version is released:

- Prepare the new AAB file which includes the changes to be made available. You need to use the same keystore/password every time you sign any new version of the app.
- Write down the summary of changes to be listed with new version for the existing/new users.
- To update the app, sign in to your Google Play console account.
- Select the app to be updated from the list.
- Tap on production and create new release.
 - Upload the new '.aab' file. This new file should be incremented by one version code number. Play store do not allow submitting any existing version of the app afresh to maintain proper versioning.
 - Provide the remarks for the update.

- The remarks will be visible on Google Play store under the What's New Section of the app.
- The remarks can tell the user what changes have been made into the new version of the App.
- Tap on Review Release for submitting the app.
 - Full Rollout: If you want to Rollout the new app update to all user segment of the app, select the rollout percentage to 100%.
 - Partial Rollout: If you want to Rollout the new app update to only specific segment of users of the app, you can change the rollout percentage e.g. update will be available to only 20% of user segment.
- Google also provides managed app release where you can schedule the release of any new submitted version of the app.

User Ratings and Reviews

- App Reviews: Users can rate your app on Google Play with a star rating and review. Users can only rate an app once, but they can update their rating or review at any time. The app ratings are in the scale of 1-5 where 5 is maximum any app can be rated.

You need to make sure that you keep an eye on the user ratings and reviews you are getting from your app's user base. This provided opportunity to the developer to improvise the app as new version by incorporating the changes as per user feedback. Also gives a fair idea of the user expectations and app functionalities.

You can check the user ratings and reviews by visiting Play Console, select the app and go to the Ratings page to view the available ratings data.

- Access to Reply to Reviews: To reply to reviews from Play Console, make sure you have the "Reply to reviews" permission. *The Centre of Competence may be requested to provide you this right for your App.* You can write one public reply for each user review of your app. You can edit your reply to a review at any time. After you reply to a user's review, they receive a push notification and an email notification. Email notifications include the following information:
 - Name of your app
 - Date of the user's review
 - User's rating & review of your app
 - Your reply
 - Link to contact you by email (using the contact email address listed on your app's store listing page)

Providing reply to any user review is a good practice which helps in creating user trust in app and helps in creating a bonding. User always feel happy on getting reply to the reviews provided on any app. Any review may be positive or negative and the answer helps user satisfaction which leads to better app rating also. It also helps in building confidence among users about the responsiveness from the developers side.

Contributors

Location	HoD	Email	IP Phone	Mobile
Himachal Pradesh	Sh. Ajay Singh Chahal	ajay[dot]chahal[at]nic[dot]in	29001	9418275076
Hamirpur, Himachal Pradesh	Sh. Vinod Garg	vinod[dot]garg[at]nic[dot]in	29211	9418109710
Himachal Pradesh	Sh. Ashish Sharma	s[dot]ashish[at]nic[dot]in	29014	9418028002
Mandi, Himachal Pradesh	Sh. Sukhdip Singh	sukhdip[dot]singh[at]nic[dot]in	29237	9988345270
Sh. Paramjeet Singh, Sh. Ravi Dhiman, Sh. Rishabh Thakur, Sh. Pankaj Rana from NIC Himachal Pradesh.				
New Delhi	Ms. Nalini Sharma Nautiyal	nalini[at]nic[dot]in	5131	9810620317
Himachal Pradesh	Sh. Sandeep Sood	sood[dot]sandeep[at]nic[dot]in	29011	9418453053
Kannur	Sh. Andrews Varghese	andrews[dot]varghese[at]nic[dot]in	33206	9447647480
Bihar	Dr. Shailesh Kumar Srivastava	sk[dot]shrivastava[at]nic[dot]in	23011	9431432962
Tamil Nadu	Sh. James Arulraj	james[dot]arulraj[at]nic[dot]in	45014	9445022255

References

App Development, API Security	TAG Diary on Mobile App Development at http://digital.nic.in TAG Diary on API Security at http://digital.nic.in
Flutter	https://flutter.dev/ https://pub.dev/ https://www.yourteaminindia.com/blog/pros-cons-flutter-app-development/ https://jsonplaceholder.typicode.com/users https://www.sqlite.org/index.html
Xamarin	https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/requirements https://dotnet.microsoft.com/apps/xamarin https://www.xamarin.com
Apache Cordova	https://nodejs.org
Web API's	https://www.c-sharpcorner.com/article/web-api-in-asp-net/ https://www.guru99.com/soap-simple-object-access-protocol.html https://dotnet.microsoft.com/apps/aspnet/apis https://RESTfulapi.net/rest-architectural-constraints/
Google Play	https://play.google.com/console/about/ https://support.google.com/googleplay/android-developer/topic/9858052?hl=en
Mobile App Development	https://www.charterglobal.com/understanding-the-3-types-of-mobile-apps-development-services/ https://www.invonto.com/insights/mobile-app-development-process/
Mobile App Testing	https://saucelabs.com/blog/choosing-the-right-mobile-test-automation-framework https://www.rishabhsoft.com/blog/mobile-app-testing https://www.perfecto.io/blog/automated-testing-vs-manual-testing-vs-continuous-testing https://www.softwaretestinghelp.com/cloud-mobile-testing-services/
Adobe XD	https://www.adobe.com/in/products/xd.html
Apple Test Flight	https://developer.apple.com/testflight/