UNIT 2 COMPONENTS OF A MOBILE APPLICATION

Structure

2.0 INTRODUCTION

Mobile app consists of various client side components and support infrastructure components. It is important to understand various components of mobile app to design and develop an effective mobile app. The knowledge about the architecture for various kinds of mobile app and their applicability would guide us to select the most appropriate mobile app for a given scenario.

Mobile app architecture also provides a perspective about the user interface design, integration aspects and other such considerations.

In this unit, we closely look at various components of mobile app and have a deep dive study of architecture of various mobile apps. At the end we will also look an end-to-end case study.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- understand key concepts of mobile app architecture,
- know the details of architecture of native mobile app,
- know the details of architecture of hybrid mobile app,
- know the details of architecture of mobile web app,
- know the details of various client side and support infrastructure components of mobile app and
- look at an end to end case study of an Android app.

2.2 ARCHITECTURE OF A MOBILE APPLICATION

Mobile applications can be classified into mainly 3 categories:

- **Browser based mobile web apps**: These are pure web applications that are designed using responsive web design techniques that can cater to variety of devices and form factors.
- **Native mobile apps**: These apps are built for specific mobile platforms (such as iOS, Android) that can fully leverage the device capability. Normally native mobile apps are built using SDKs provided by mobile platforms.
- **Hybrid apps**: These apps can be developed using web technologies such as JavaScript and can also partially leverage the device capabilities using a web to native layer.

Table 2.1: Provides the key differences of three types of mobile app architectures

	Browser based responsive mobile web	**Native apps**	**Hybrid apps**
Development technology	Usually uses responsive web design (RWD) using CSS3, media queries	Native apps are developed using iOS, Android and other mobile platforms	Developed using a single technology (usually JavaScript) and can be ported to any native technology
Features	Only used for responsive and interactive UI. Cannot fully use all mobile device features. Provides Best Portability, Time to Market, maintainability	Can fully exploit all device capabilities such as camera, sensor, contact books and such. Provides best user experience	Limited usage of device native features
Applicability	Can be used for information display without any offline support and for hosted solutions	Can be used for features needing rich user experience, user engagement such as games. Can be used for offline requirements and for features needing device specific features	Can be used for easier portability across mobile platforms and cross-platform compatibility. We could use this for faster time to market scenarios with frequent releases.
Compatibility	Works across mobile devices and platforms. Offline not supported.	Native apps run on specific mobile platforms; offline capabilities supported	Hybrid apps can be ported to various mobile platforms

Development	Fast development and maintenance	Higher cost	Lower development cost due to portability. Build once deploy to any platform model.
Advantages	Standards compliance, lesser development cost.	High performance, rich UI, can exploit native mobile features,	Easy portability
Performance	Relatively lower performance	Optimal performance	Relatively lower performance
User experience	Provides optimal experience for web experience	Provides excellent UI features leveraging the device capabilities to the fullest extent	Provides decent UI experience.
Hosting	Cloud or on premise platforms	App stores or marketplace	App stores or marketplace

2.2.1 Architecture of a native Mobile App

Native Mobile app fully utilizes the device capability. Figure 2.1 and Figure 2.2 depict architectures for iOS and Android native apps:

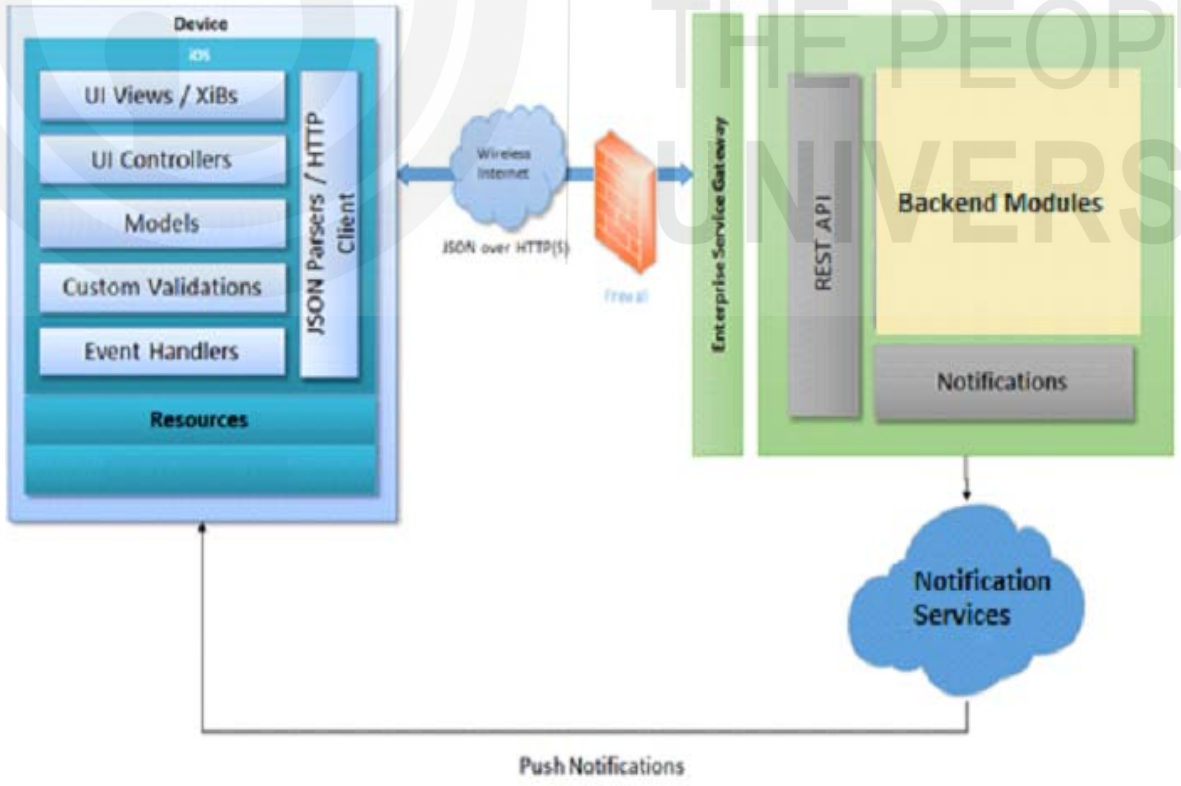

Fig. 2.1: iOS based native Mobile App

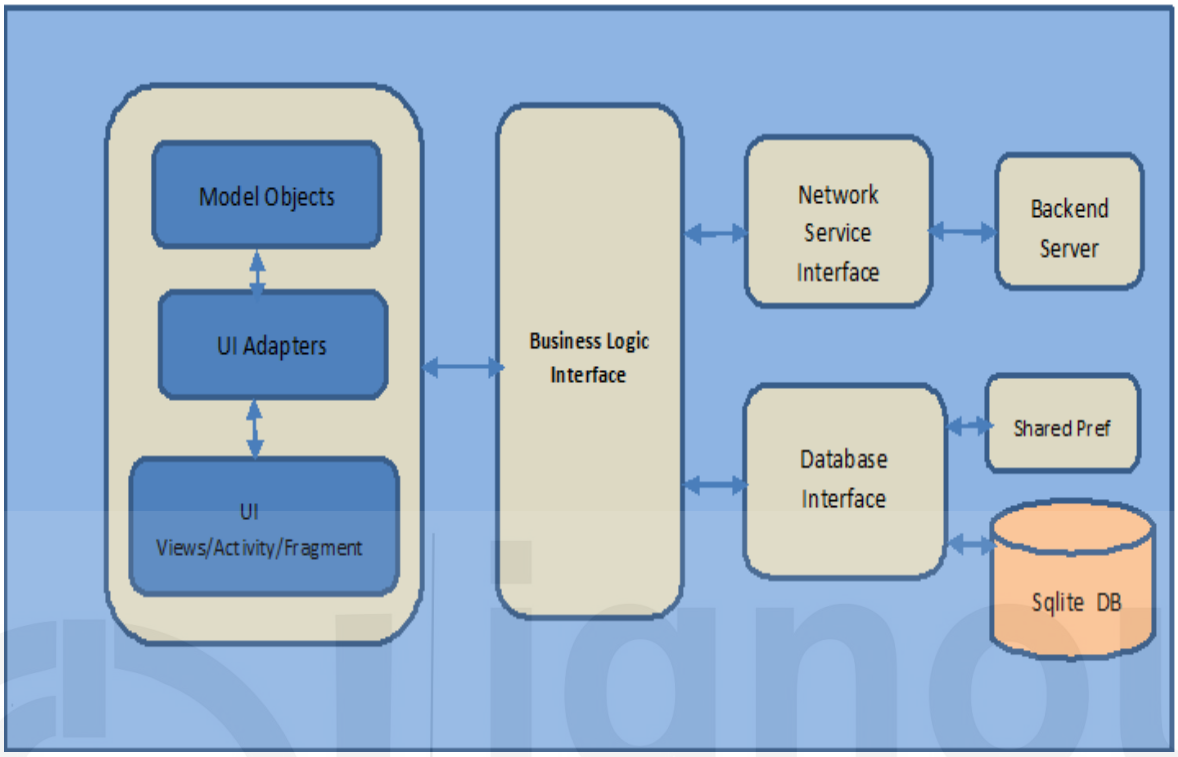

Fig. 2.2: Architecture of Android based native Mobile App

2.2.2 Architecture of a hybrid Mobile App

Hybrid application targets different mobile OS with a very thin application shell with a web view. One time application development is needed with publication to various mobile app stores. Hybrid apps are very popular in developing Single Page Application responsive application for Tablets and Smart Phones.

The key components of the hybrid app are given in Figure 2.3. The Mobile application development using Hybrid app development has the following benefits:

- **Usability:** Ability to use device specific features to improve usability
- **Maintainability:** Easy maintenance of code and ease of future enhancements
- **Extensibility:** Support for multiple platforms
- **Device Diversity**: No additional effort for supporting new devices
- **Portability:** to various mobile platforms
- **Faster time to market:** through quicker deployment to mobile app stores.

We can have mobile specific services which consolidate the required backend services and provides optimized data for the mobile application.

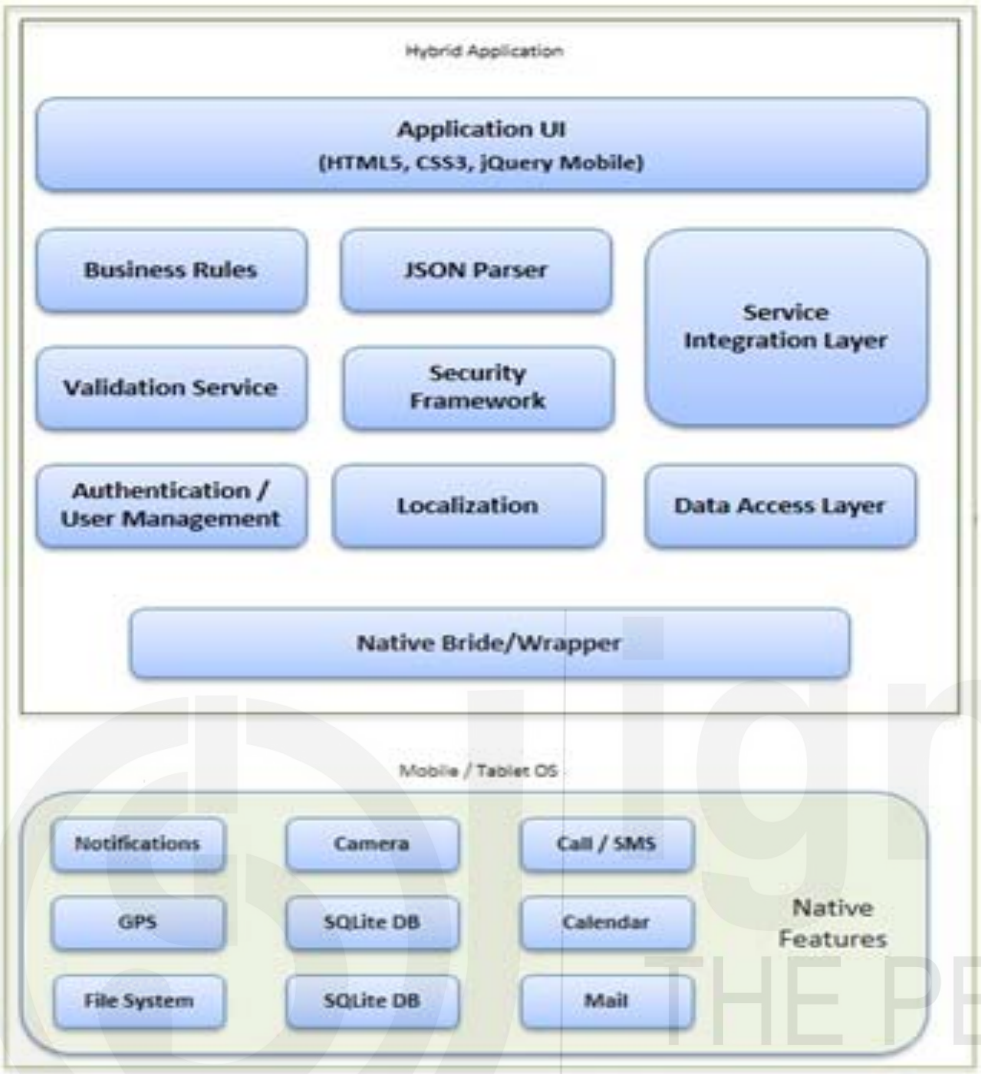

Fig. 2.3: Architecture of a hybrid Mobile App

Figure 2.3 is an expanded view of the mobility solution. It details out the functionalities and the interfaces with the underlying components as given below:

Application UI: All the screens would be developed using HTML5, CSS3 and JavaScript's. User experience would be specifically designed for Mobile Apps with targeted platform compatibility.

Business Rules: are operational procedures for making decisions, identifying workflows and operational choices.

Service Integration Layer: Integration of all the business services with the backend system would be managed by this layer.

Security Framework: All the data persisted within the device will be stored securely using the encryption techniques specific to each platform. Random generated unique key will be used for encryption and decryption. The key used to encrypt will persist in the device securely. Advanced Encryption Standard (AES) is the algorithm used for security purpose. The data sent through the network will be specific to the backend service.

Authentication / User Management: The user specific data management and the business logics will be handled by this layer. The authentication will be done in online / offline mode.

Localization: This is the layer which will provide the multi-lingual capability to the app. All the static literals specific to each language will be maintained locally inside the app. The literals specific to the dynamic content would be fetched from the backend services.

Data Access Layer: All the data stored locally would be accessed through this layer. This consists of data objects and the local storage. SQLite would be used as local storage for the application.

Native Bride/Wrapper - This layer will provide the capability to access the devices native features. This is the combination of the mobile core platform and JavaScript.

The end to end flow of the hybrid mobile app is shown in Figure 2.4 and the corresponding layers explained.

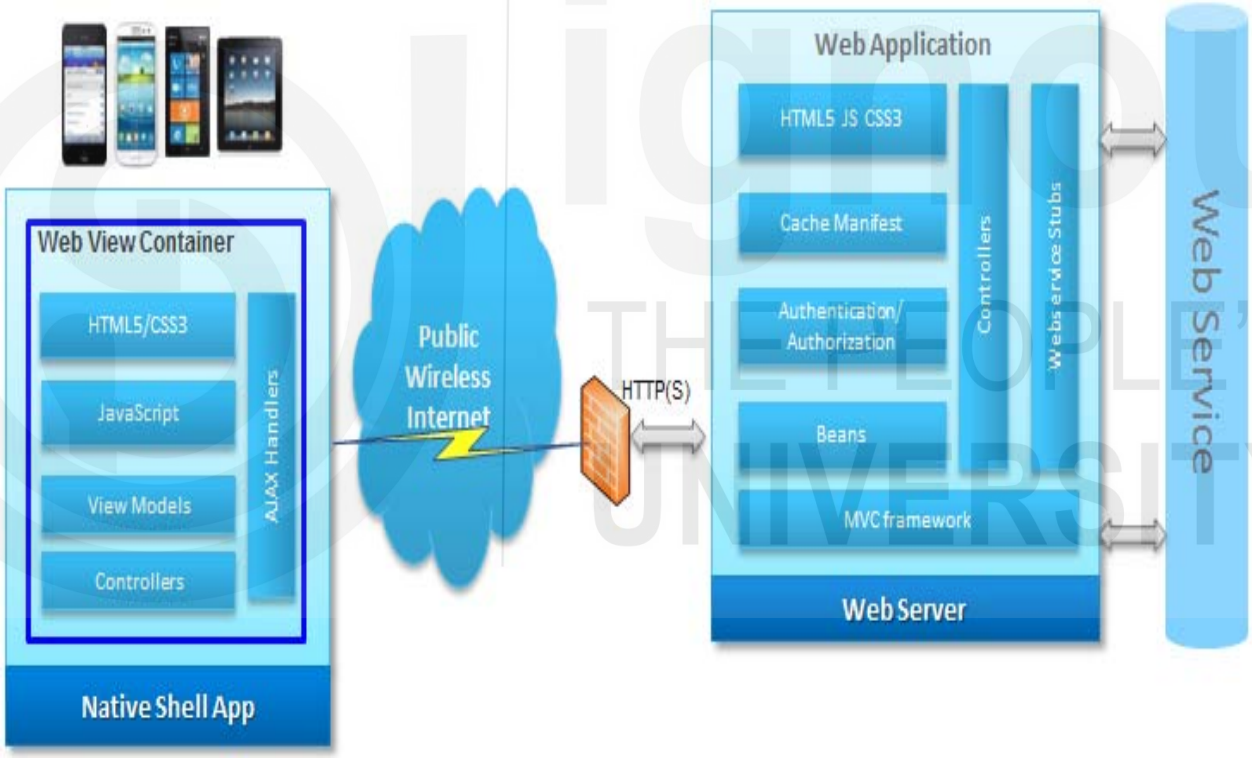

Fig. 2.4: End to End Flow of a hybrid Mobile App

Presentation Layer

- UI will follow a Responsive Web Design (RWD) and will be developed using HTML5.0, CSS3, Twitter Bootstrap with JQuery
- The UI can be packaged using Apache Cordova to create a Hybrid App. REST/JSON Web services will be leveraged.
- Hybrid apps will have two components, the native component and the web component.

Business Layer

- REST/JSON Web services will be leveraged to access server side applications.
- Business layer consists of server side components for Services Integration and other custom business services.

Security Layer

- Authorization to the app user will be leveraged with user registry services.

2.2.3 Architecture of a Mobile Web App

Mobile web apps are mainly developed using responsive design. RWD provides flexible and responsive layouts that automatically adjust themselves for various devices and form factors which include the following techniques:

Fluid Design

contains the page size in relative units.

Flexible image and media

Flexible images would automatically scale based on the screen resolution.

CSS3 media queries

would provide flexible layouts and use various device dependent CSS style rules.

Figure 2.5 shows the responsive layout displayed over multiple devices.

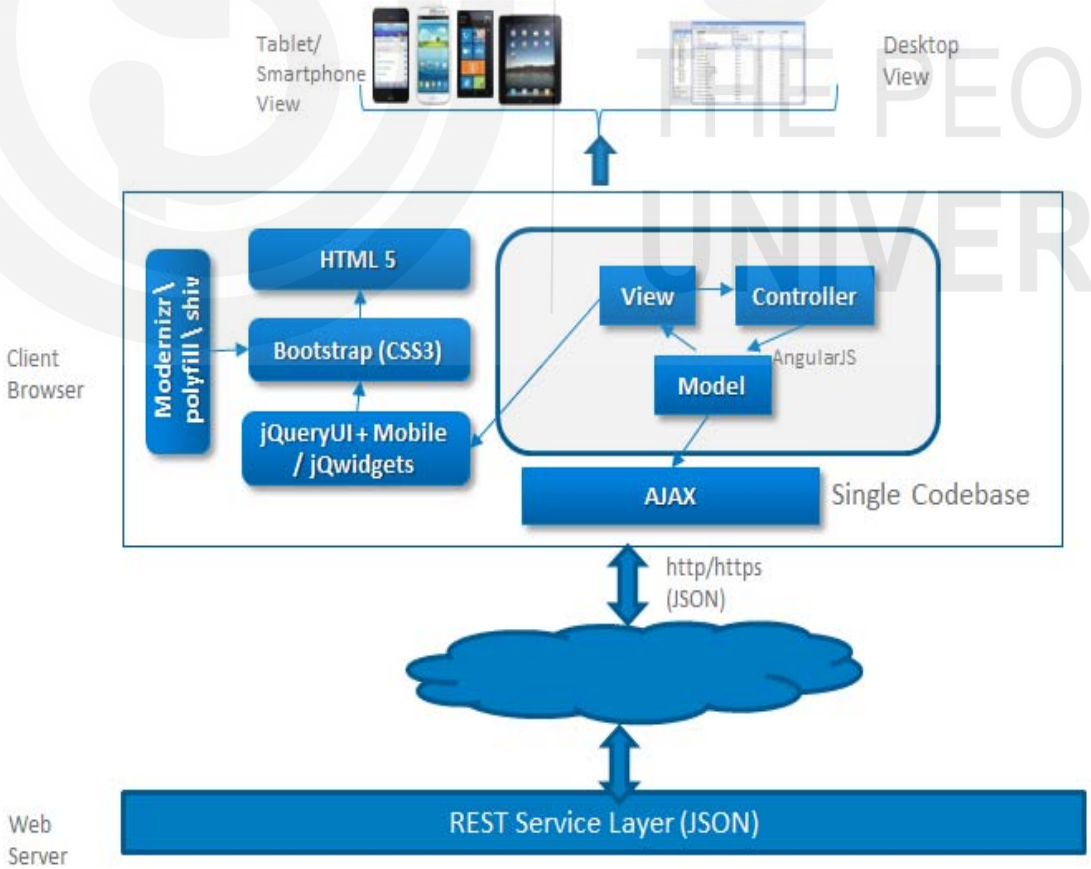

Fig. 2.5: Mobile web app using RWD

The salient features of RWD are given below:

- User Experience - a responsive website is flexible and will adapt screen layout to provide optimal user experience.
- Cost Effective - only one version of the source code and only one content management system to update the content which means that RWD will save both time and money.
- Improved SEO - The URL structure will remain the same on all devices, improving your search engine visibility and rankings. Instead of building links or optimizing content for multiple websites, you will only need to market a single responsive website.
- Increased Conversions - The potential customer can access your website with ease from their preferred device enhancing user experience that will increase sales and improve conversion rates.

The key architectural principles followed while designing a mobile web app are as follows:

- Open standards based technology.
- Layered architecture using MVC pattern
- Modular and extensible component design
- Adoption of services oriented architecture for integration
- Leveraging open-source technologies wherever applicable
- Performance based design
- Continuous build and integration approach for execution

A sample mobile web app architecture with responsive components in a MVC architecture is depicted in Figure 2.6:

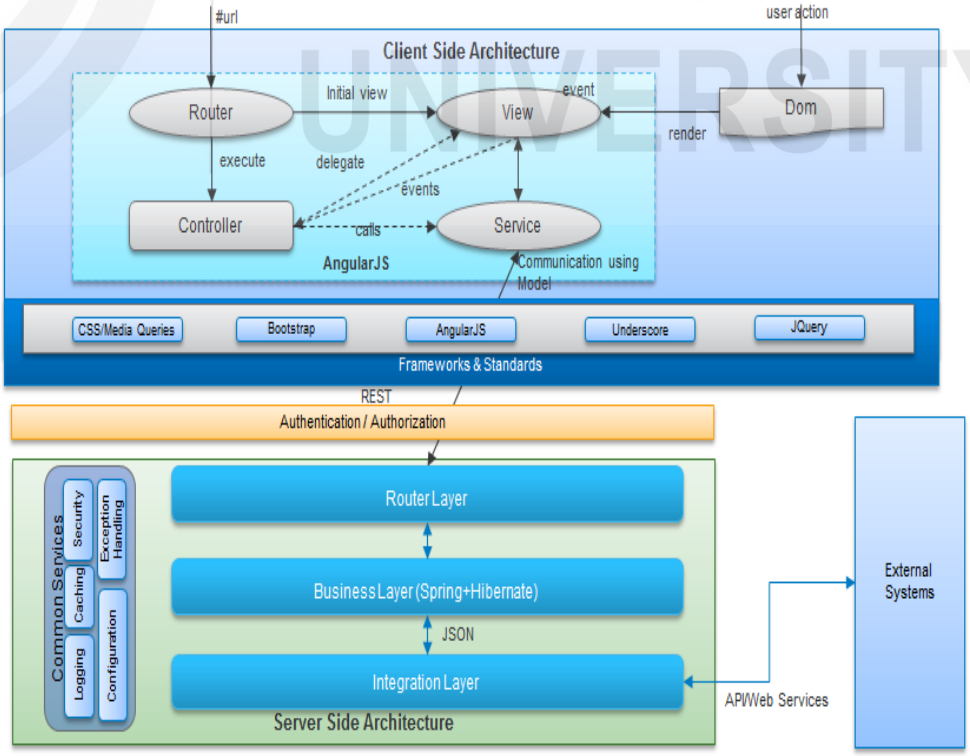

Fig. 2.6: MVC architecture of mobile web app

The MVC architecture depicted above comprise of building server side and client side components where client components will have presentation logic based on MVC design pattern using approved JS (Java Script) libraries whereas server layer will have all the business logic which will be exposed through RESTful services. Client components will interact with server side components using REST services to fetch and persist the data.

Client Layer

- **MVC**: Client-side MVC framework (Angular JS) will be used to structure the application, which will provide:
  - routing support within the application to allow navigation in application;
  - event-driven interaction between views and model;
  - Simplified CRUD (Create, Read, Update and Delete) invocations on RESTful services.

Server Layer

- A router layer maps the URL with a service end point. This layer will be an interface with client side. These REST services will pass the request parameters to business layer.
- Business layer will validate the input data, perform any data manipulation or business logic implementation on the data retrieved from the Integration layer.
- Integration layer will basically integrate with the external system which will be exposing APIs/web services. Data communication will take place using JSON (Java Script Object Notation) format.

Roles of various layers are depicted in Table 2.2:

Table 2.2: Layers in a Web Mobile App

Layer	Client/Server	Description
Router	Client side	Linking URLs to controllers and views.
View	Client side	Consists of Angular directives, templates, CSS and JS files.
Controller	Client side	Controller ties the view with back end data.
Services	Client side	Services interact with back end data.
Router	Server Side	Router maps the URL with service end point.
Business	Server Side	Takes care of business rules, validations, data transformation and such requirements.
Integration	Server Side	Integrates with external system which will be exposing APIs/web services.

A sample technology stack for implementing the mobile web app is depicted in Table 2.3:

Table 2.3: Technology Stock for Mobile Web App

Technology to be used	Usage side	Description	Benefits
AngularJS	Client side	MVC based JavaScript framework	Introduced clean MVC design pattern for client side web applications Built-in dependency injection subsystem Extensible using directives
Bootstrap	Client side	HTML and CSS based front end framework	Makes front-end web development faster and easier
HTML/CSS	Client side	Standard technologies for web development	
Spring	Server side	REST API backend built using Spring or Node.js	Extensive and Scalable framework
REST Services	Server side	Web services based on REST (http/https) URLs	Lightweight, maintainable, and scalable

2.3 COMPONENTS OF MOBILE CLIENT APPLICATION

The following are key components of mobile client application:

- UI components such as widgets, buttons, screens and navigation components
- JavaScript libraries in case of mobile web applications
- Widget libraries to manage the client-side widgets
- UI controllers for client side MVC frameworks
- Validators and event handlers for handling interrupts and notifications
- Model components for handling the data
- Service interface layer to access services
- Light weight database to persist the user data.

2.4 COMPONENTS OF MOBILE SUPPORT INFRASTRUCTURE

The following are main components of mobile support infrastructure are:

- Mobile device management (MDM) is used for configuration and management of mobile devices. MDM is mainly used for distribution of mobile apps, enforce policies. Other key functions of MDM software are:
  - Asset management and device grouping
  - Remote software management
  - Configuration management
  - Performance diagnostics and health check of devices including memory, battery, network information with reporting capabilities
  - Restore and backup of device data such as calendar, contacts, notes and such.
  - Application and service provisioning
  - Firmware upgrades
  - Logging and reporting
  - Troubleshooting
- Mobile security management for enforcing security policies
- Mobile middleware for acting as adaptors and to expose services. Mobile middleware components are also responsible for data transformation, routing, caching, governance.
- Other mobile infrastructure elements are:
  - Spectrums for 3G and 4G

☞ **Check Your Progress 1**

1) ……. Apps are normally built using SDK.

2) Mobile web apps pre-dominantly use …….. technique to cater to multiple browsers and form factors.

3) …… in Android app makes backend service.

4) Most commonly used light weight DB in a mobile app is ……

5) In MVC framework, …… glues view data with back end data.

2.5 END TO END CASE STUDY OF AN ANDROID MOBILE APP

The following is a detailed case study of an end to end flow of an Android mobile app for retail domain. The detailed case study would help us to understand various components and integration mechanism for building a native mobile app. The high level architecture of Android App is given in Figure 2.7.

The following are key requirements for the app:

- App should scan the bar code for a retail solution and provide mapping support

- App should support multiple languages
- App should support data encryption during transmission
- App should support synchronization features.

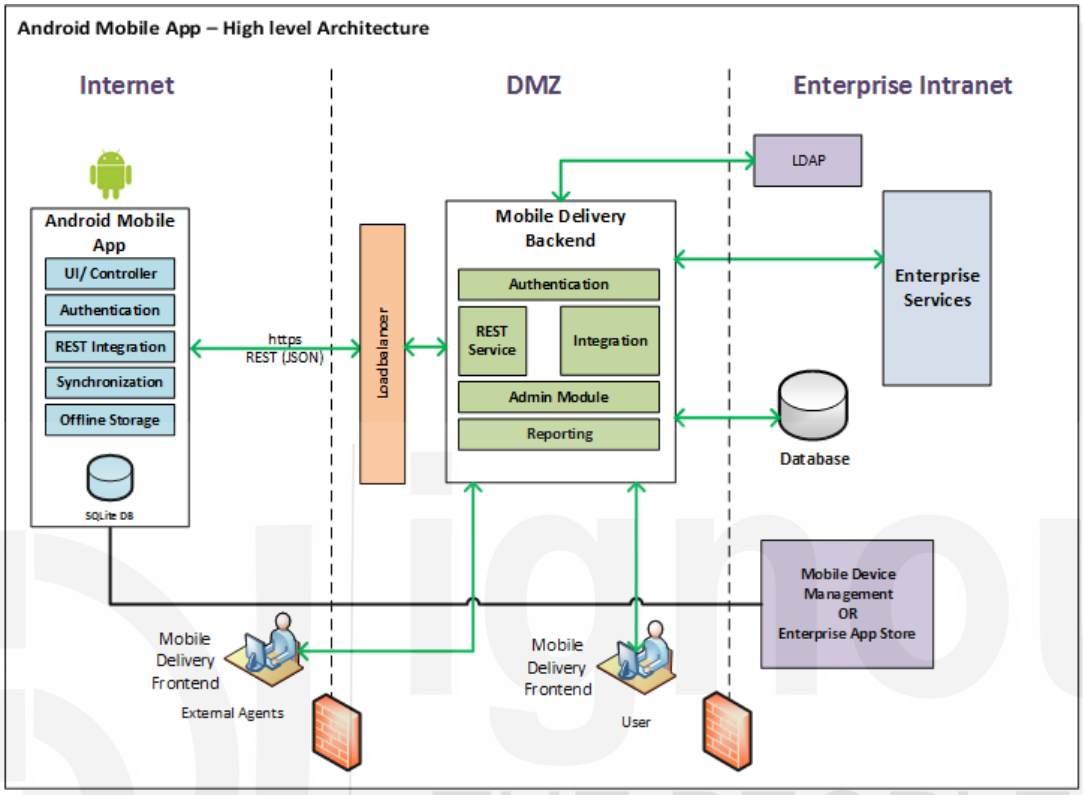

Fig. 3.7: Android App High level architecture

Different components of Android App are described below:

Client Tier

- Native Android application– to work on Android Phones
- Solution uses encrypted storage to securely store sensitive data in device. Data can be stored in the SQLite data store for access in offline mode.
- Application supports multi-language screens. The application will be designed to use language strings that will be added in the properties files. Depending on the logged-in user's device language, the appropriate language string file is loaded.
- Application leverages the device camera to scan the barcodes.
- Application will leverage native device libraries to support features like Tracking by providing GPS coordinates in latitude/longitude format
- Application will use standard google map APIs for all map specific workflows

Client Integrations

- Application authentication would be done against the Mobile Backend system.
- Application would integrate with the RESTful services exposed by the Mobile Delivery Backend system. Data transmission format will be JSON.

- The communication channel will be secured by using https for all data transmission
- The application uses offline data storage. The application expects the backend system to provide delta data retrieval (for offline sync). The delta data retrieval is needed to reduce the amount of data transmitted across network.
- The data synchronization would be implemented as a manually initiated sync. This is to provide more control to the user and to prevent periodic polling to the backend (which would increase battery usage on the device).
    - Last synchronized time and number of entities modified on the device which are pending to be synchronized with the backend may be displayed in the header. This will provide a better view on the data freshness to the users.

Figure 2.8 depicts various components in the backend of Mobile App.

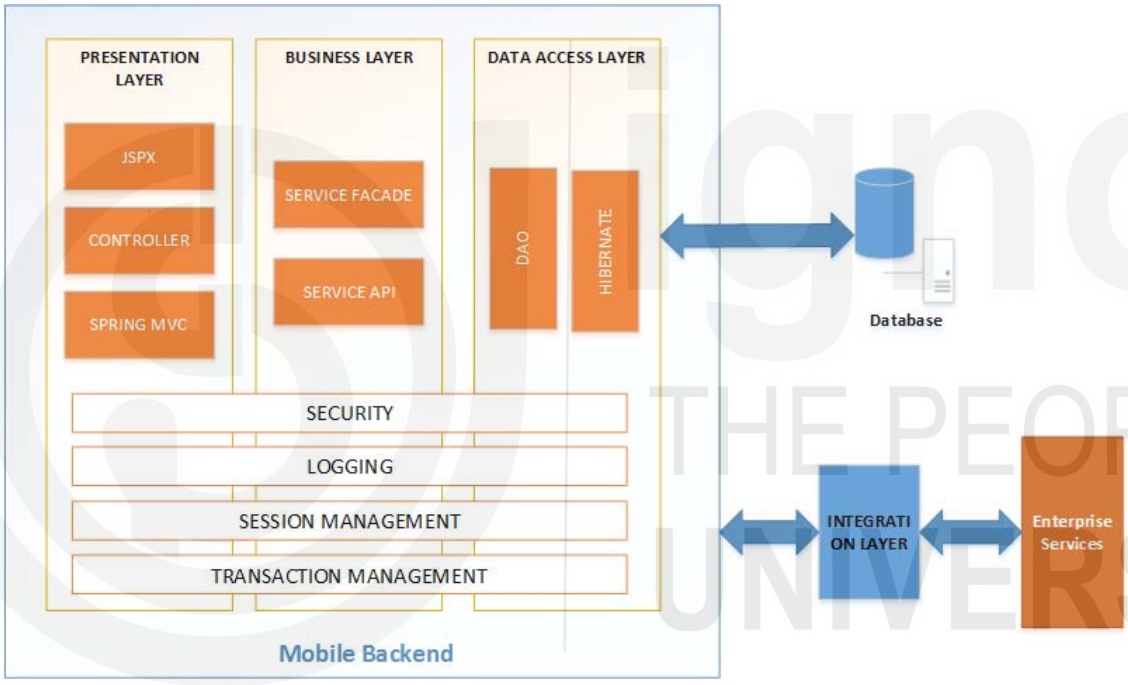

Fig. 2.8: Mobile backend components

The components are described below:

- JSPX web pages developed for Mobile Platform
- Spring MVC for attaining loose coupling between components
- Business Layer that comprises of “Service Façade” and “Service Implementation” patterns
- Data Access Layer would comprises of DAO (Data Access Objects) and would connect to database using Hibernate
- Spring Framework would be used to cater for Logging, Security, Session Management and Transaction Management
- Integration Layer would provide interface with the Enterprise services
- REST services would be exposed to Mobile client Application

☞ **Check Your Progress 2**

1) The communication channel will be secured by using …… for mobile app.
2) Offline data sync can happen through …….. data retrieval.
3) …….. is used for configuration and management of mobile devices.

2.6 SUMMARY

In this unit, we started discussing the high level architecture of various kinds of mobile apps. We then detailed layer-wise components of native moble app, responsive mobile web app and hybrid app. We looked at various components of mobile client application and support infrastructure. We also detailed the roles and technology stack for responsive web app. We saw an end to end case study for Android app.

2.7 ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 1

1) Native Mobile Web
2) RWD
3) NetworkService
4) SqLite DB
5) Controller

Check Your Progress 2

1) HTTPS
2) Delta
3) MDM

2.8 FURTHER READINGS

References

http://en.wikipedia.org/wiki/Mobile_security

http://csrc.nist.gov/publications/drafts/800-124r1/draft_sp800-124-rev1.pdf

https://en.wikipedia.org/wiki/Mobile_architecture

https://en.wikipedia.org/wiki/Mobile_asset_management

https://en.wikipedia.org/wiki/Mobile_device_management

https://en.wikipedia.org/wiki/Secure_Mobile_Architecture