LESSONS LEARNED MINI 2

Android Components & Android Studio

- Overview of Android Architecture : Android has different components such as Android Runtime, Frameworks, Linux Kernel.
- Learnt how to incorporate SDKs and Emulator
- As part of assignment 1, I learnt the complete lifecycle of an Android Activity and the methods used to implement those lifecycle such as onStart(), onCreate(), onResume() and so on.

User Interface Design Practices

User Interface is a crucial aspect as it needs to be intuitive

- Design in XML
- Using strings, colors, res and layouts, used **declarative approach** where all the IDs and string values were given in strings.xml and style.xml
- Supporting multiple screen resolutions by generating **drawables as vectored images** and putting them in the corresponding hdpi, xhdpi, mdpi folders. Created different layouts for normal screen sizes and tablets (sw600dp)
- Exception Handling, using inbuilt support as well as creating custom exception
- Fragments and Fragment lifecycles and **why to use fragments**. In case of large screens, it is important for the screen to not be left empty and instead pop up with a new action in the remaining space. In such cases we use fragments. Apart from this, Fragments can be used to implement **gestures and actions** as well
- UI controls can be anything like a button or a menu activity **and onClickListeners** and the likes can be used to get data from them. Every control lives in the context of a layout
- The purpose of an **Intent** : Invoke a UI component, show a screen/raising an event
- **Register content in Manifest file** as the Manifest file recognizes all activities and permissions required for an app to run
- **Menus are not considered good UI practice**, menu is for application functions which are used less frequently
- **Bundle** is a an efficient way to pass data between activities

APIs

- Media APIs to contain videos and music and play them like rewind, ff and so on
- Telephony APIs
- Location Services : Mapping API provided by Google and API to handle GPS which is the Location API. Learned the **difference between MapView Class and MapActivity Class**.
- Understood how to get frequent refresh and updates of GPS data by using LocationManager's **requestLocationUpdates**
- Learned APIs pertaining to Bluetooth, NFC, Sensors

Android Security Model **implements permissions** in the Manifest file and requires that all application packages need to be **signed digitally** before being installed in the Play Store

Lessons Learned during Individual Assignments

Assignment 1 :

- Introduced to Activity LifeCycle , learned about Android Development LifeCycle

Assignment 2 :

- We implemented Mortgage calculator app. In this I learned how to get data from the different input fields and this was like an introduction on using Android. Also learnt about using Content Providers like SQLITE

Assignment 3 :

- Performed a basic analysis of student records and learned about media player APIs and Containers like Gallery View and so on

Assignment 4 :

- Learned about Location Services

Lessons Learned during Project Implementation

Most of the work done in the project borrowed heavily from the **concepts learned in Mini 1**. We learned how to incorporate and use Abstract Classes, Interfaces while developing the application

Asynchronous Tasks

We understood the **differences between Background Services and Asynchronous tasks**. The reason we learned about them was because we had to have a clear picture about when to use each of them. When there is **no need for user interaction, we can use a Background service**. If we want to abstract certain activities that are **happening within the application while still presenting an apt user interface** to the user and maintaining screen presence, we can **use asynchronous** tasks

We understood the **bind API** associated with services, the **doInBackground API** associated with Asynchronous tasks and used Asynchronous tasks in our project to connect to remote services

As an extension from Mini 1, we used **HTTP GET** requests to request for services from the Back End via these tasks.

Connecting to content provider from the Android Application

We designed and implemented a tiered approach for our application. As part of storing data, we executes SQL queries on a normalized database which we designed ourselves

- **Presentation Tier** : Interacts with the users
- **Business Logic Tier** : Gets components from the user interface and puts into the objects in the front end which implement the logic
- **Integration Tier** : These are typically local and remote services which help us contact the MySQL content provider. At the front end we have services implemented as Asynchronous tasks and at the backend we have

- **Content Provider Tier** : These resources provide mechanism to store user data. It can be databases like SQLITE or MySQL

We connected our servlet to the backend by providing **10.0.2.2 as the IP address** to connect to while testing on the Emulator . While testing on a physical device, we provided the **static IP address** of the machine which runs the server.

Working with libraries

During the course of this project **we improved code usability** the passing of objects from the Front End to the Back End by using serialization libraries like **JSON and GSON as compared to making our class implement the Serializable interfaces** as in Mini1

We learned to choose **GSON over Jackson** as we are implementing Web Services and GSON has functions like **toJson and fromJson which makes implementation easier.**

Designing using XML

Although Android Studio provides drag and drop functionality for the purpose of UI design, for fine grain tuning of UI, we grasped some of the properties for the different UI tools and widgets. We also learned how to implement **styles and themes**

API

Some APIs that we learned during the implementation of this project pertained to the UI class of Expandable Views and SMS and Telephone Managers. Ex :
The getChild method returns the tab which has been selected by the user.

- The getGroupView method returns the parent (in our case this is a user's name) which was selected by the individual using the app.
- We learnt how to invoke a pre-installed app by passing Intents. Ex : We used ACTION_VIEW intent to check if an SMS can be sent by the default application

REFERENCES/ACKNOWLEDGEMENTS

1. We referred to example code pertaining to DatePickers in Android Tutorials Point
2. We referred to http://www.veereshr.com/Android/AndroidToServlet and developed a similar model to connect to the Android device/emulator
3. We referred to the Best Development Practices document by Aniket Rao and Pushkaraj and implemented a similar design