Android app for “CHARUSAT e-NoticeBoard”

Prepared by
Umang.J.Patel (16IT092)

Under the supervision of
Prof. Hemant.N.Yadav

A Report Submitted to
Charotar University of Science and Technology
for Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology
in Information Technology

IT345 Software Group Project-II (5th sem)

Submitted at

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY
Chandubhai S. Patel Institute of Technology
At: Changa, Dist: Anand – 388421
May 2018

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

This is to certify that the report entitled "Android app for CHARUSAT e-Noticeboard" is a bonafied work carried out by Umang.J.Patel (16IT092) under the guidance and supervision of Prof. Hemant Yadav for the subject Software Group Project-II (IT345) of 5th Semester of Bachelor of Technology in Information Technology at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfilling the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,

Prof. Hemant Yadav,
Assistant Professor,
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Parth Shah
Head & Associate Professor
Department of Information Technology
CSPIT, Changa, Gujarat.

Chandubhai S Patel Institute of Technology

At: Changa, Ta. Petlad, Dist. Anand, PIN: 388 421. Gujarat

ABSTRACT

The primary purpose of doing the project was to improve the circulation and distribution of notices in educational institutes like schools and colleges. The need for this project was felt after facing this problem first-hand in my university. The announcements and events are currently circulated via e-mail, which cluttered the mailbox, leading to students not able to get the relevant and desired info at the correct moment.

CHARUSAT e-NoticeBoard is a large project involving different software modules being integrated with each other. In this project, my contribution is the Android app where students can view the notices and interact with an intuitive chatbot where they can ask some questions and get info on the app itself.

I was involved in working on data communication between the REST API (deployed in the project) and the Android app. The key challenge to such data transfer is speed and reliability. To help me cope up with these issues, I utilised a 3rd party library called Retrofit, which aids in transfer of data from JSON format to POJO objects resulting in easier and error-free data modelling compared to other 3rd party libraries.

Moreover, the app utilises modules of a modern Android app architecture called Android Jetpack. This architecture is recommenced by Google itself, and helps in building robust and modular Android apps. Along with using this architecture, old and less effective techniques had to be reworked, which was quite helpful for learning new concepts in the Android world.

ACKNOWLEDGEMENT

I, Umang.J.Patel, have the made an Android app for a larger project called as "CHARUSAT e-NoticeBoard" under the guidance of Prof. Hemant.N.Yadav. This project has led me to new beginnings in how design and coding go hand in hand while developing a high-quality app. When I started to make this app, it took me some time to deeply understand and design the flow of the user interaction. As I progressed, I asked for reviews from a few colleagues so as to make the app more usable for the users and help me improve my plan of action for this app ahead. I thank Prof. Hemant.N.Yadav for his guidance that led this app to its correct destination through the app development procedure.

TABLE OF CONTENTS

INTRODUCTION

Project Overview :-

CHARUSAT e-NoticeBoard Android app is an Android app which displays notices pushed by Faculty on Android devices. Users can even ask their questions through an intuitive chatbot and get relevant information on the go.

The app fetches data (notices) from a REST API deployed on “CHARUSAT e-NoticeBoard” and displays it on the screen. Users can even search notices based on its titles and can help find the right info on the go.

The REST API is consumed using a 3rd party library called Retrofit. Retrofit helps to deserialize JSON data into a suitable Java POJO class. After deserialization, it gets stored in a list and this list is displayed on the screen using suitable UI widget called RecyclerView.

The users can also ask queries of general information about the CHARUSAT university via a chatbot screen (a part of the larger “CHARUSAT e-NoticeBoard” project).

Scope :-

It is software system, which provide the functionality of notifying the students about the events, and other information by the faculties without using the mail system in an android app installed in the students’ smartphones. The android app also contains the CHATBOT functionality through which users can get the general information about the CHARUSAT University and their respective department. The chatbot is also to be deployed on the CHARUSAT currently developing website. All notices which are posted by the faculties that were also displayed on the CHARUSAT website with the help of a plugin so that users can get notified by visiting that website, in case of they don't have Android App. But there is a problem for iPhone users because the iOS app is not developed for this project and in future we will develop the iOS application so that iPhone users can also be notified about any events and other information.

Objective :-

The main objective of this project is to notify the CHARUSAT students with the latest information and events without any mail system or going to any physical noticeboard.

SYSTEM ANALYSIS

User characteristics :-

This e-Noticeboard System is divided in three parts. First, one is the front-end part that is the website in which the notices will displayed. Second, one is the back-end part through which the faculties or any other administrator user can post the notices and that is stored in the database and with the help of plugin, notices will displayed on the website. Third, is the android app through which the students or the users can be able to see the latest information or notices and that notices will fetched with the help of REST API in JSON format.

Android User can only get the notices if there is web service exist through which is data from database is transferred to android phone. The REST API in JSON form transfers that information. The notices (stored in JSON format) will be deserialised using Retrofit library and will be displayed on the screen.

Tools and Technologies :-

1) Android Jetpack

A developer told me to use Android Jetpack to start the app development because it was easier to learn and provided solutions to common developer pain points such as activity lifecycles, separation of concerns etc.

Jetpack is a set of libraries, tools and architectural guidance to help make it quick and easy to build great Android apps. It provides common infrastructure code so you can focus on what makes your app unique.

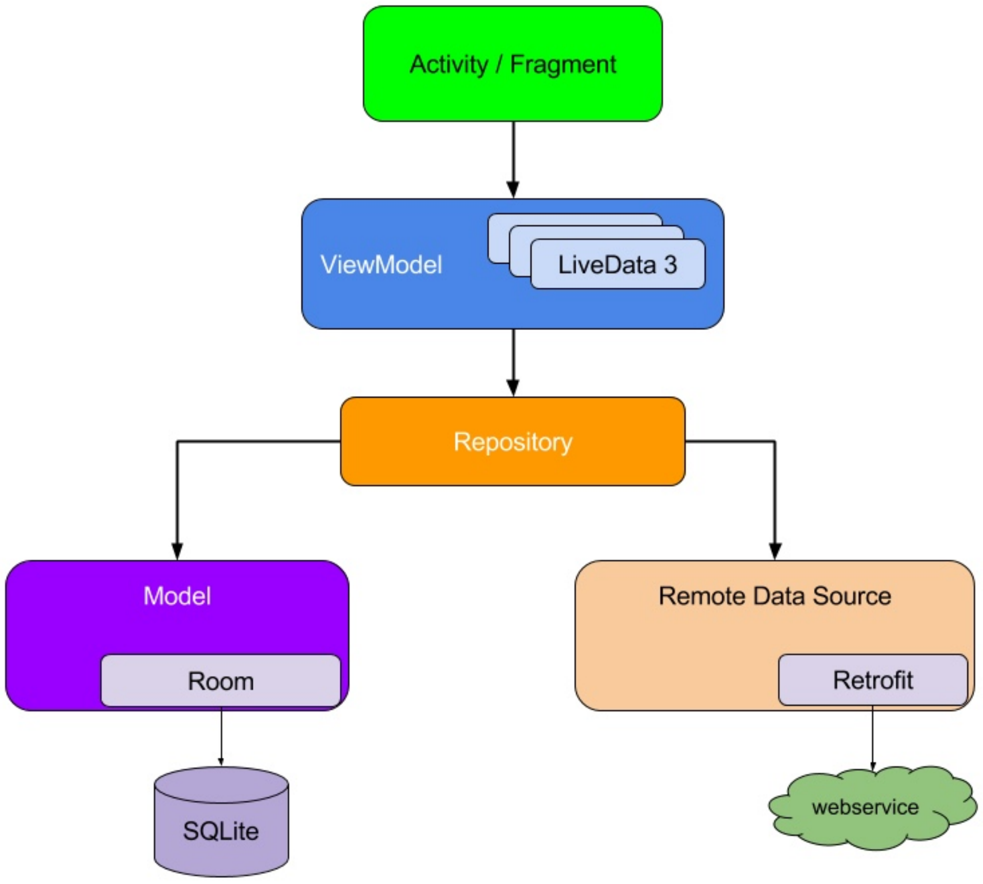
• Accelerate development
Components are individually adoptable but built to work together while taking advantage of Java and Kotlin language features that make you more productive.

• Eliminate boilerplate code
Android Jetpack manages tedious activities like background tasks, navigation, and lifecycle management, so you can focus on what makes your app great.

• Build high quality, robust apps
Built around modern design practices, Android Jetpack components enable fewer crashes and less memory leaked with backwards-compatibility baked in.

To start learning it, I turned to its well-written documentation allowing me to jump into developing basic apps.

After feeling confident enough, I began writing the app as per the recommended app architecture as follows :-



Fig, 3 : Recommended Android app architecture

• Activity/Fragment

It is the view visible to the user on the device screen. You need to make layouts using XML (Extensible Markup Language) for providing widgets such as Buttons, EditText, TextView, etc.

• ViewModel with LiveData

A ViewModel provides the data for a specific UI component, such as a fragment or activity, and handles the communication with the business part of data handling, such as calling other components to load the data or forwarding user modifications. The ViewModel does not know about the View and is not affected by configuration changes such as recreating an activity due to rotation.

LiveData is an observable data holder. It lets the components in your app observe LiveData objects for changes without creating explicit and rigid dependency paths between them. LiveData also respects the lifecycle state of your app components (activities, fragments, services) and does the right thing to prevent object leaking so that your app does not consume more memory.

- Repository

Repository modules are responsible for handling data operations. They provide a clean API to the rest of the app. They know where to get the data from and what API calls to make when data is updated. You can consider them as mediators between different data sources (persistent model, web service, cache, etc.).

- Room

Room is an object mapping library that provides local data persistence with minimal boilerplate code. At compile time, it validates each query against the schema, so that broken SQL queries result in compile time errors instead of runtime failures. Room abstracts away some of the underlying implementation details of working with raw SQL tables and queries. It also allows observing changes to the database data (including collections and join queries), exposing such changes via LiveData objects. In addition, it explicitly defines thread constraints that address common issues such as accessing storage on the main thread.

- Remote Data Source

Basically it is your backend provides a REST API, Firebase, etc. used to fetch data from the internet.

2) Retrofit

Retrofit is a REST Client for Java and Android. It makes it relatively easy to retrieve and upload JSON (or other structured data) via a REST based web service. In Retrofit you configure which converter is used for the data serialization. Typically for JSON you use Gson, but you can add custom converters to process XML or other protocols. Retrofit uses the OkHttp library for HTTP requests.

To work with Retrofit you need basically three classes.

- Model class which is used to map the JSON data to
- Interfaces which defines the possible HTTP operations
- Retrofit.Builder class - Instance which uses the interface and the Builder API which allows defining the URL end point for the HTTP operation.

Retrofit library makes it easy to consume REST APIs over the internet and helps the developer to build functionalities quickly.

3) Android Studio:

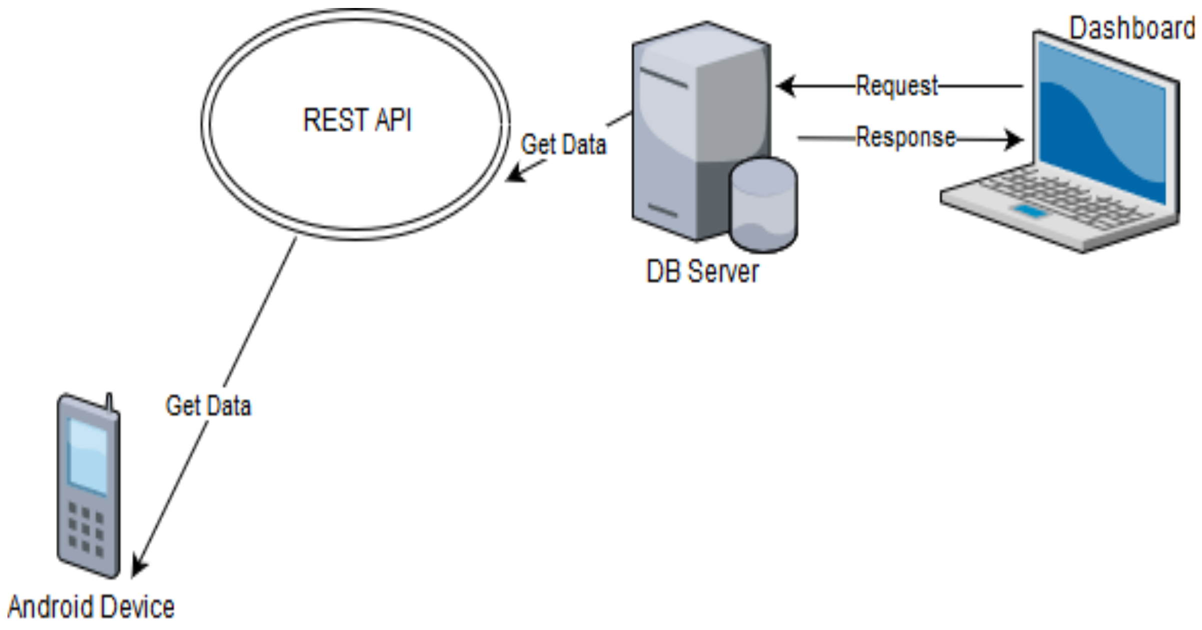
Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code

- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging, Firebase and App Engine.

SYSTEM DESIGN

Flow of system :-

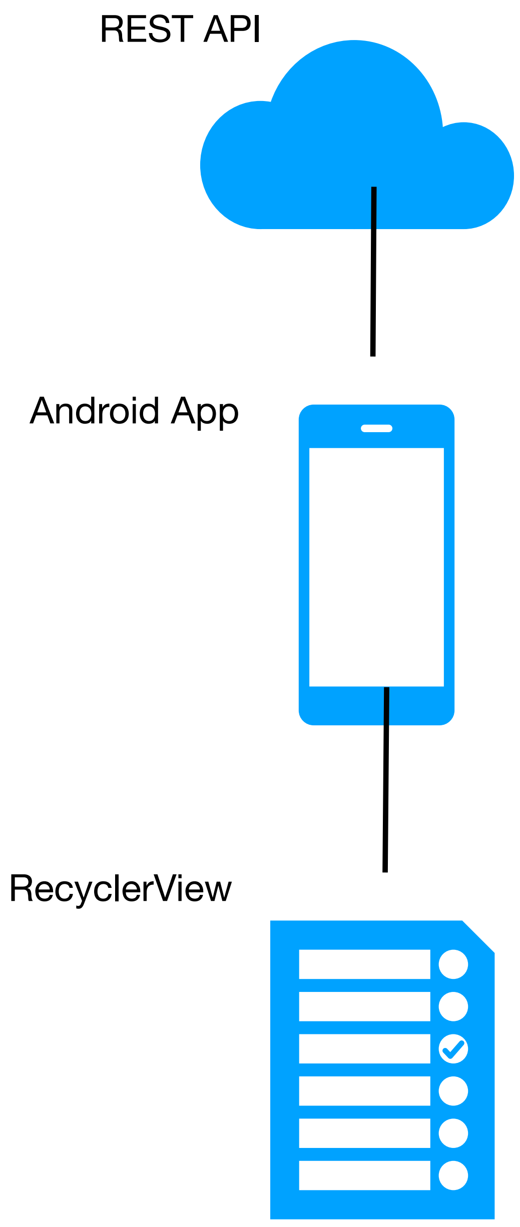


As shown in above figure, dashboard can create the notice, delete the notices, update the notices and read the notices. That notices are stored in the database serve having the table 'notices'.

The REST API is connected to database server with the table 'notices' so that it can get the data and transfer this data to android app.

REST API gets the data from notices table and transfers the data in form of JSON format and after processing that JSON format data. The notices are displayed in the android app with help of Retrofit networking library.

Flow of working :-



The JSON data is consumed from the REST API using the Retrofit networking library to the Android app. After deserializing this data to Java POJO format, the notices are stored in a LiveData object. This live data object pushes the notices whenever it observes a change in its data. This LiveData object then is used to display the list of notices using RecyclerView.

IMPLEMENTATION

Implementation Environment:-

To create the best possible mobile app, I planned out the development process into three parts.

1. Designing the app.
2. Writing the app.
3. Testing the app.

Designing the app:

Mobile app development is a very complex process. You require proper thought process and critical thinking for making your app loved by your users. Before writing any code, you need to plan your code. Planning your code is not as difficult as it looks like. I began prototyping and wire framing the app with only a pencil and paper. This method allows you to be much more creative and artistic as you start to visualise your ideas and put it on the paper. making some progress even though you feel it’s a bad idea. The more I drew, the more I learnt about difference wire framing techniques and even discovered some new ones.

The app flows is the most important part of the prototyping process. Your app must have less amount of screens for a particular action like sharing an image or purchasing an item on an e-commerce app. By remembering this single fundamental principle, I made sure to create least amount of flows for an action in my app.

During the design process, I planned the code in the back of my mind. I started to think how could I do this in my app. So I started writing some bullet points for performing an action i.e wrote the pseudocode.

For example, fetching a notice follows a few steps as follows :

1. Check whether device is connected to the internet.
2. Fetch JSON data from the REST API using Retrofit.
3. Store it in LiveData object.
4. Show the list on the screen.

Doing this allowed me to explore the documentation and find a suitable solution without requiring any code to write or modify. This also enabled me to find additional features that can improve the performance of the app.

Firstly, to follow the “separation of concerns” principle, I divided my classes based on the screen. Naming packages as per the screens helps in easier code navigation during the coding process and change code faster than ever.

LiveData is an observable data holder class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state. This helped me to write less boilerplate code regarding data observation in the activity.

Data is fetched from the database to the RecyclerView in device via ViewModel and LiveData.

Module Specification :-

1) REST API Client

The Retrofit library requires the developer to create an interface wherein you annotate a method with the API endpoint you desire to consume. It is shown as follows :-

```
public interface NoticeService {
    @GET("/api/notices")
    Call<List<Notice>> getNotices();
}
```

To use this Java method, we need tp create a Retrofit client which will call the above method. It is shown as follows :-

```
public class NoticeClient {

    private static final String BASE_URL = "http://
172.16.11.70:8010/";

    private static Retrofit getInstance() {
        return new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory
.create())
                .build();
    }

    public static NoticeService getNoticeService() {
        return getInstance().create(NoticeService.class);
    }

}
```

We need to add a GSON Converter class so that the JSON data can be converted to Java POJO objects.

2) Master-detail screen

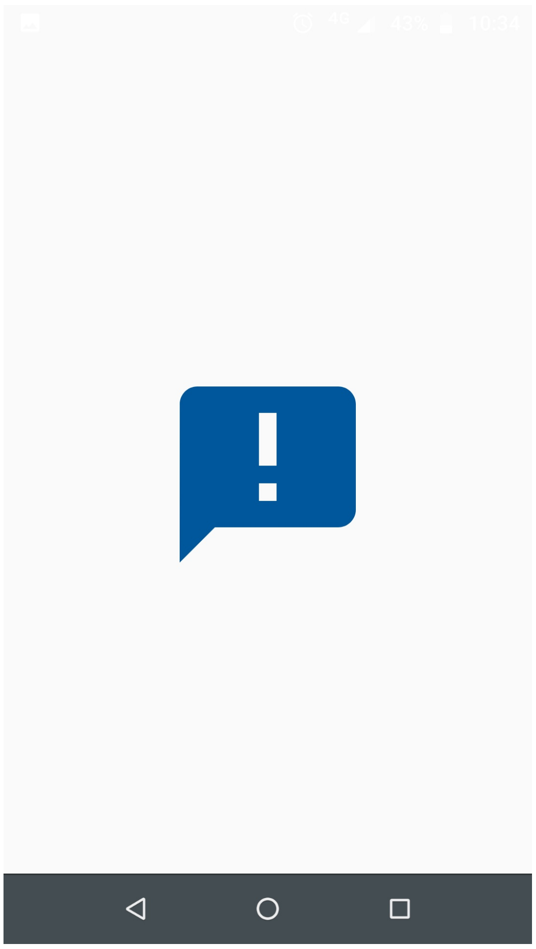
The list screen comprises of a UI widget called as RecyclerView which displays a list of notices.

Now whenever a list item is clicked, it needs to parse this data to a detail screen wherein detailed view of notice is displayed on the screen.

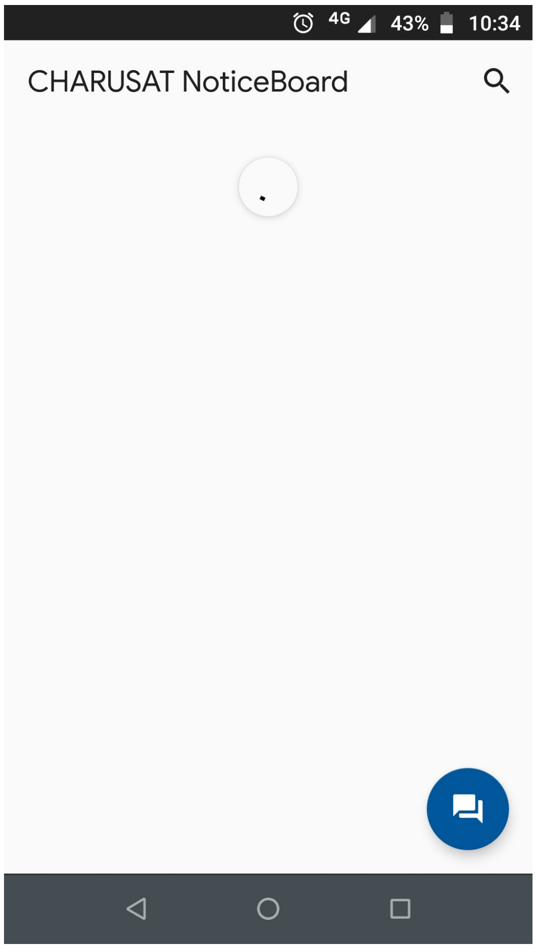
The following code does the job of parsing the individual notice data on to a detail screen :-

```
@Override
    public void onClick(View v) {
      itemView
     .getContext()
.startActivity(DetailActivity.newIntent(itemView.getConte
     xt(), mNotice));
}
```

Snapshots :-



Splash screen



Notice List screen

CONSTRAINTS AND FUTURE ENHANCEMENTS

The limitation of the current working of the app is that the notices are displayed only using the WiFi network on the CHARUSAT university. This can be resolved by deploying the API on a domain.

Moreover, the search functionality is constrained only to searching the notices via its titles. The search can be improved by providing a filtering screen where users can filter notices based on dates, faculty who issued the notice, department, etc.

CONCLUSION

This report leads you through the journey of how I as a student tried to solve a real world problem that can lead to better life just using mobile technology. Throughout the development of this project, my perspective had to be changed from a developer's one to the user-oriented one. I had to create a mobile app that the users can find it easy to understand and use. A lot of learning took place to reach to the correct destination. I was challenged but I stood up and ended up developing a robust, high quality, production app. Throughout the process, a lot of blogs, documentation and video tutorials enlightened me with key points that were helpful to speed up the development process. Thanks to Mr. Hemant Yadav without whom this project could not be much better than its initial versions.