Android Development

by

Md. Tariq Hasan
15101141

An internship report submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2023

© 2023. Brac University
All rights reserved.

Declaration

I hereby declare that

- This defense report, entitled Android Development, is based on my original work conducted during my internship at FountainIT & Software Services.
- This report has not been submitted in whole or in part for any other degree or qualification.
- I affirm that this internship was carried out in accordance with the ethical standards of my university, and that all internship procedures were approved by the relevant defense committee.
- The workings and features presented in this report are based on the projects performed and analyzed by me during my internship, and any opinions expressed are my own.
- Any sources used in this report have been acknowledged and properly cited.
- I attest to the accuracy of the information presented in this report, and I am prepared to defend it before the examining committee.

Student's Full Name & Signature:

Md. Tariq Hasan
15101141

Approval

The internship report titled "Android Development" submitted by

1. Md. Tariq Hasan (15101141)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 31, 2023.

Examining Committee:

Primary Supervisor:
(Member)

Arif Shakil
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The IT industry has reached its peak growth in this century and continues to grow exponentially with the perpetual rise in demand for better and unique accessible products and services throughout the entire globe. This has forced researchers and developers of profit-making firms to pay keen interest in maximizing profit particularly with the supply of mobile devices for its ever-increasing customer base. Large supply even with a marginal profit led to huge gains in revenue and this economic philosophy is openly and widely implemented by the continuous development of smartphones.

With such a rapidly growing industry, developing smartphone apps has become a new phenomenon in the IT sector and Google's Android operating devices on mobile handsets continue to reign the maximum number of users of the target population. Such potential is indeed grasped by like-minded people and this has led to this internship to follow through.

The objective is to master the basics of a programming language (Java), a framework to make development easier and compact (Android) and implementation of the newest and robust User Experience and Performance libraries that Google has already provided for exchange of no expense or monetary value. This brought many challenges that made me spend hours on a single feature/ bug during my internship. The 6 months that I spent during my internship happens to be one of the best learning experiences that I ever had and I continue to cherish those moments that led me become a software engineer in the present.

Acknowledgment

I would like to begin by expressing my deepest gratitude to Almighty Allah for giving me the strength and guidance to complete this internship.
I would like to extend my sincere thanks to Sumon Ahmed, Tech Lead, and Bazlur Rashid, DevOps, of FountainIT for providing me with the opportunity to work and learn in a professional setting. Their expertise and encouragement have been instrumental in shaping my learning experience.
I am also grateful to my university primary supervisor, Arif Shakil, for his invaluable guidance, support, and feedback throughout this internship.
Lastly, I would like to thank my parents for their unwavering love, support, and encouragement throughout my academic journey. I am deeply grateful for their sacrifices and guidance, which have enabled me to reach this point in my career.
Also, I am indebted to all those who have supported me in this journey, and I hope to use the knowledge and skills gained from this experience to contribute positively to society.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Preamble

Fountain IT & Software Services is committed to provide professional support to your company or person with our experienced and certified experts & using our latest technological tool sets. After a couple of months of furnishing my java and Android Studio essentials, I was able to make mobile android applications from scratch. Of course, those were small and simple apps consisting of a maximum of 3 to 4 screens, each performing simple tasks such as user authentication, showing data from local database in scrollable lists, simple UI designs and programming. Right after that I was engaged into the live project of my company where I would design UI from Figma files and code them in .xml files and connect to Android's internal libraries for utility in java. The front-end (Views/Screens/Layouts/Drawable) took considerable amount of time.

1.2 Challenge

Initially all designs were hardcoded, meaning all attributes were separately coded into each screen. This caused code duplicates, more time to reciprocate and replicate code to different screens. Worst of all, for example, if the font of a single text were to change due to client's requirements, I would have to manually hunt down all the textViews (Views containing texts) of the entire project and change each and every one of them by hand. This would become frustrating and quite impossible to maintain if the size of an app became large; hundreds of changes would have to be made just to change a single element's attribute and a moderate sized app has more than a thousand.

1.3 Learning

Hence, learning proper UI and programming guidelines from Google's official support blogs, tutorials and Google's material design UI websites became the primary goal to maintain the scalability of an application. And time investment and hard work gradually ensured that the designs were not hardcoded but was put in a separate .xml style tag which was called from yet another separate theme tag that governed the themes and styles of the entire app. All in all, the best UI principles portrayed in

Google's android development documentations, training and GitHub resources were followed to the letter. This led me to design and develop a large-scale app of more than 20 different screens, all of whose toolbars, fonts, colors, styles and attributes could be changed by changing it in just one place.

1.4 Document Objectives

This document is prepared out of necessity to complete my internship in BRAC university. Throughout this phase of the report, I will display my contributions and various implementation of Google's best practices in designing an android app. Android itself has numerous open-source libraries which are free to use and minimizes the urgency to create custom libraries and views. This will also include many built-in widgets, tools and libraries that comes readily with Android Studio and other 3rd party library providers on GitHub.

Chapter 2

Background

2.1 Company's Profile

Overview

Fountain IT & Software Services are dedicated to providing professional support to their clients with the help of their experienced and certified experts who utilize the latest technological tools available. The company takes pride in their ability to share their knowledge and expertise gained from numerous past engagements and experiences, and they continuously evaluate and improve their skills through self-assessment. Additionally, they do not hesitate to decline projects that are beyond their scope of knowledge and experience. The company experts are carefully selected based on their proven professional skills in multinational companies and their certified knowledge in the subject matter. Customer satisfaction is one of their core values, and it is reflected in their company policy.

Mission & Vision

The company is technology-focused and aimed at the next generation, and their vision is to create value for their customers' businesses by offering cutting-edge technology solutions through their expertise and experience. Their mission is to take enterprise management to the next level by incorporating machine learning and artificial intelligence into their services.

2.2 Services

Web Design and Development Professional web designers and developers in Bangladesh can help businesses generate greater revenue leads. They optimize website design to increase the chances of attracting clients and enhance scalability, ensuring a professional and polished end result.

Digital Marketing With the support of enhanced mobile infrastructure and low data pack rates, internet users are increasing day by day, especially on small-screen devices. Big IT companies like Google, Facebook, and Amazon provide valuable services, making people habituated to online interactions. Their digital marketing

services help clients take advantage of these trends and reach their target audience effectively.

Cloud-based Apps Development Industry demands have shifted towards cloud based software development, as cloud solutions ensure high availability and less maintenance for end users. Their team is equipped to provide any type of web-based or cloud-based solutions as per client's specific requirements.

Web Application Development Today, every application needs a web-based component and must be responsive to run on different devices. Their team can help clients understand their requirements, transform them into technical specifications, and develop customized web applications that meet client's unique needs.

2.3 Products

Framework for Discovering and Reconciling Telecom Network Physical Assets Telecom and communication service provider networks are complex environments with multiple vendors and technologies, making it difficult to manage physical assets for optimal utilization, reporting, and valuation. Their framework offers a proven solution to automatically discover all active physical assets from vendor and network management systems, and reconcile them with client's financial asset management system database. Additionally, an optional workflow engine can manage all asset lifecycle types within the solution.

Case Diary: Lawyer Case Management Application Case Diary is a mobile and web application for legal professionals that allows for easy access to case records from anywhere, eliminating the need for bulky physical diaries. It helps lawyers and litigants remember upcoming case schedules and find case details quickly. Every day, thousands of cases from different courts are uploaded to Case Diary, reducing the hassle and effort of collecting data. Users can easily add their own cases as well.

MasterMoshai: Distance Learning Platform MasterMoshai is a distance learning platform integrated with Zoom online video conferencing, focusing primarily on advanced IT training. It offers learning by doing with real-world projects and hands-on exercises that lead to real skills mastery. An active forum enables learners and instructors to communicate and solve project problems through one-to-one sessions after completing the course.

AttendR: Attendance, Roster, and Payroll Management Application AttendR is an application that allows for roster configuration in different shifts and automatic generation of payables from hourly pay rates. With just a fingerprint device and a computer, attendance can be taken and, based on configuration, automated daily pay rates can be generated for employees. Users can configure/copy rosters, manage timesheets, attendance, pay rates, generate reports, and more.

E-learning Solution Complete web-based e-learning solution enables IT learning for educational institutions.

E-commerce Solution E-commerce solution allows for online product sales, with users able to upload detailed product descriptions by category and subcategory. Monthly sales reports can be generated as well.

Accounting Management Online accounting software is designed for small businesses to manage finances and stay on top of their cash flow. Invoicing, accepting online payments, and keeping track of expenses couldn't be simpler.

Event Management System Event management system enables event managers to create event calendars that attendees can view and sync with their calendars. The solution provides event registration, roles, attendance, and invitations with group/individual email options.

Point of Sale (POS) Sales and inventory management system provides role-based functionalities to track purchase, sales, customers, suppliers, billing management, and generate different reports.

Hotel Management System Web-based hotel management system allows hotel managers to handle all hotel activities online. An interactive GUI and ability to manage bookings, rooms, inventory, accounting, and more provide flexibility and convenience for users.

Employee Tracking & Attendance Management System (ETAMS) Android based attendance management solution, ETAMS, is connected to a web-based backend that can be operated either through intranet or internet, depending on configuration.

Chapter 3

Internship Experience

I had never worked on Android development before and this tempted me a bit at first. Without any prior development experience, I still had some confidence as Fountain IT & Software Services was interested in making native android applications purely in Java language.

3.1 The first steps

The first step was to install Java JDK and Android Studio. Android apps are primarily developed and programmed in Android Studio: the software developed by Google.

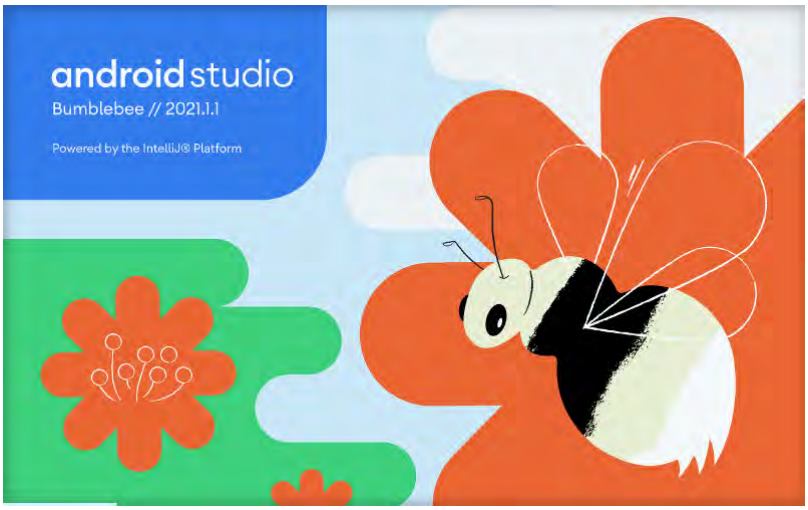


Fig. 3.1: Android Studio Splash Screen

The first steps of learning android development was to go through a youtube playlist designed by freeCodeCamp.org. FreeCodeCamp is by far one of the leading software training companies in the current world in the field of IT. Google's Android has a dedicated website for learning android development. It primarily includes material.io and developer.android.com.

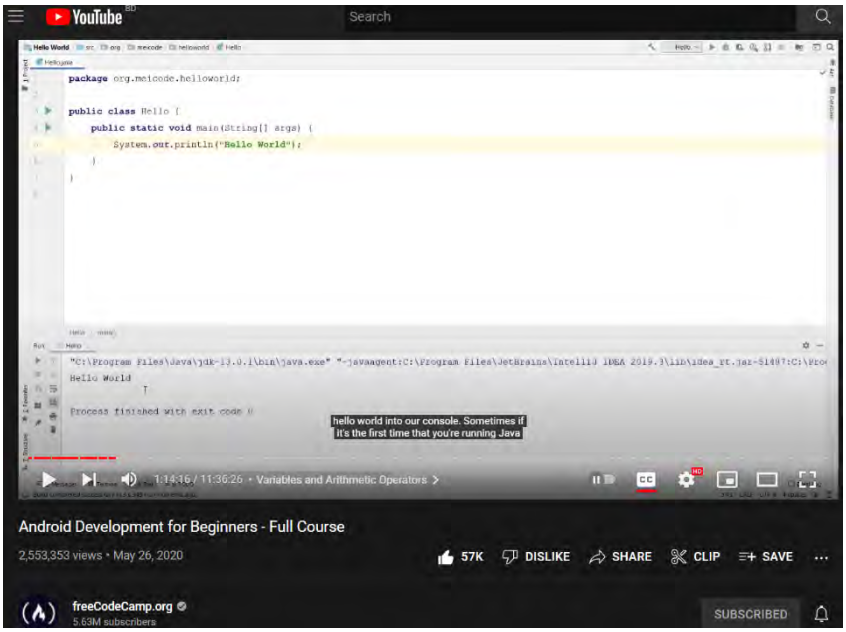


Fig. 3.2: A screenshot of a YouTube programming video tutorial

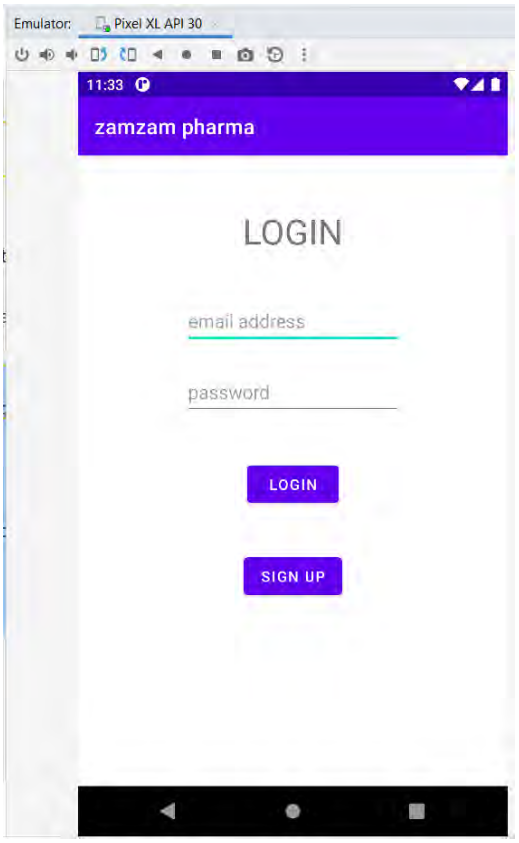
Having refreshed the fundamentals of Java programming, it was time to learn android layout design and implementing them to a mini app. In android development, it is quite unlikely and rare for a developer to code all of the logic and classes by himself. Rather, Android Studio itself provides a huge collection of libraries for developers to import and implement their functions and classes for their personal use.

I have worked with RecyclerView, SQLite, GoogleMaps SDK, Facebook SDK, Glide, Google Firebase, and other in-built and third party libraries as a beginner. Android Studio provides a built in Android Emulator for running android applications on the computer. This can also be bypassed and run directly onto the phone that is connected to the computer.

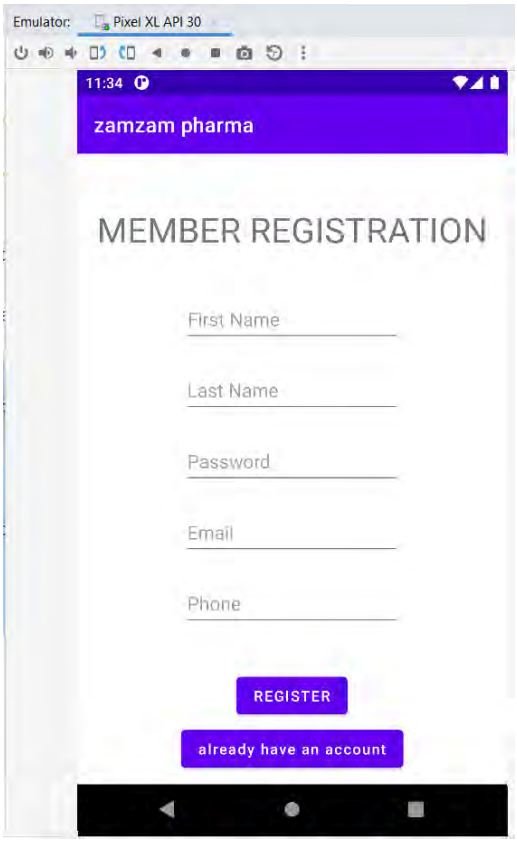
Here is a glimpse of some sample apps that I had built as a novice android developer.

3.2 Inventory Management App

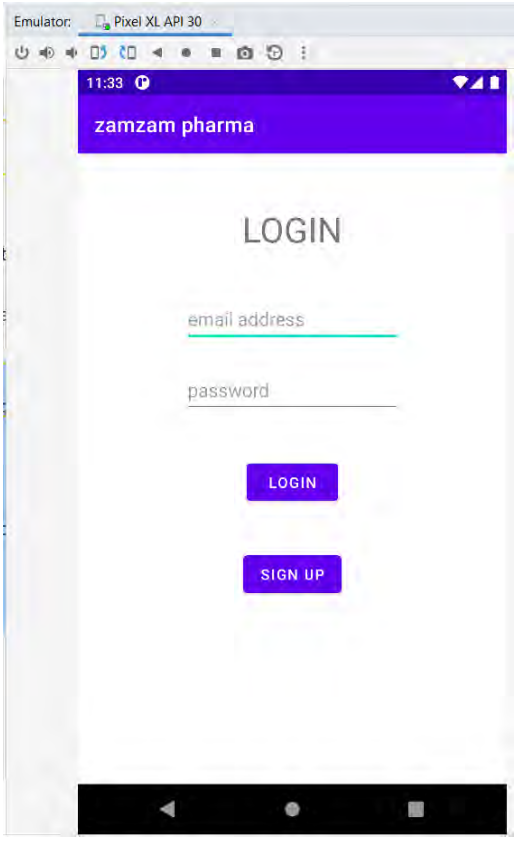
This application is a simple Data management app for a pharmacy. It had a login form for authentication using SQLite. It saves new medicines from users into local storage of the phone and displays them in a RecylerView.



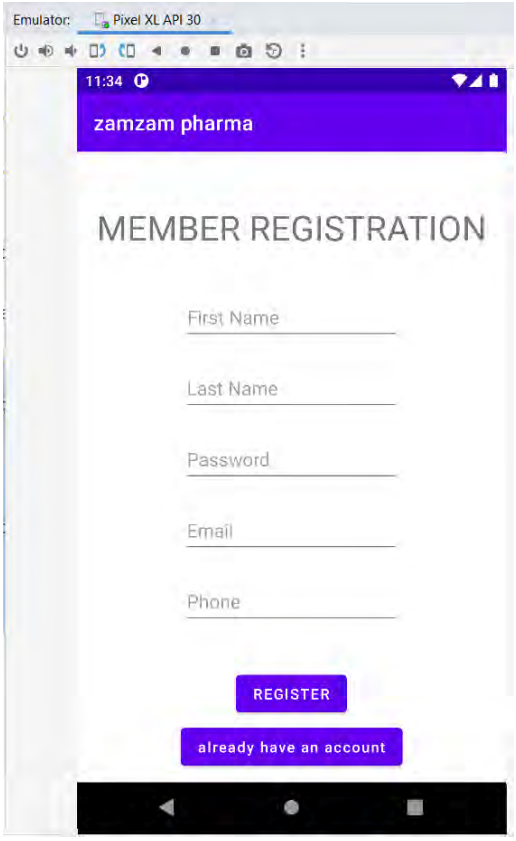
(a) Login Screen



(b) Registration Screen



(c) Dashboard

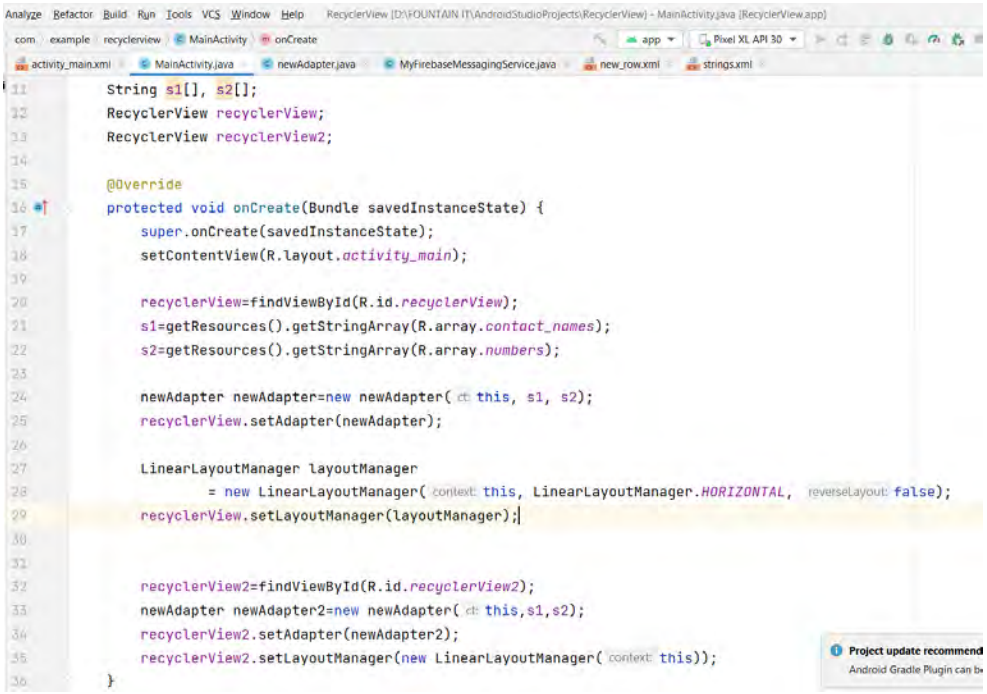


(d) Inventory List

Fig. 3.3: Sample screens of the Inventory Management App

3.3 Contacts App

This application is a basic contact list created using SQLite and RecyclerView library. The first row is scrollable horizontally while the rest of the layout is the other.



```
String s1[], s2[];
RecyclerView recyclerView;
RecyclerView recyclerView2;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    recyclerView=findViewById(R.id.recyclerView);
    s1=getResources().getStringArray(R.array.contact_names);
    s2=getResources().getStringArray(R.array.numbers);

    newAdapter newAdapter=new newAdapter( ct: this, s1, s2);
    recyclerView.setAdapter(newAdapter);

    LinearLayoutManager layoutManager
            = new LinearLayoutManager( context: this, LinearLayoutManager.HORIZONTAL, reverseLayout: false);
    recyclerView.setLayoutManager(layoutManager);


    recyclerView2=findViewById(R.id.recyclerView2);
    newAdapter newAdapter2=new newAdapter( ct: this,s1,s2);
    recyclerView2.setAdapter(newAdapter2);
    recyclerView2.setLayoutManager(new LinearLayoutManager( context: this));
}
```

Fig. 3.4: Sample code snippet

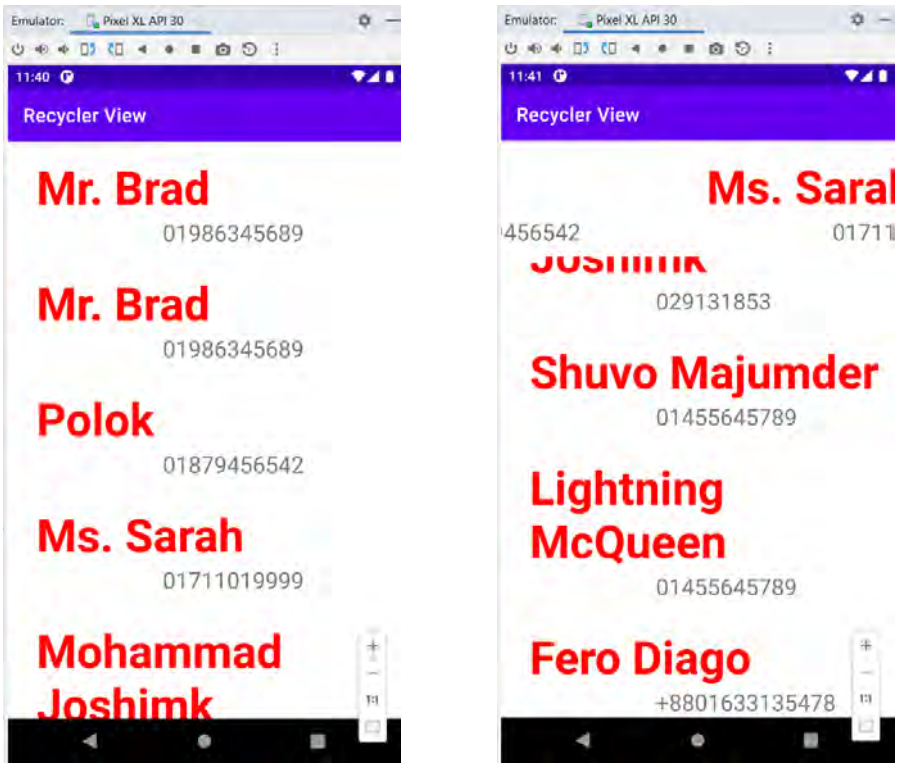


Fig. 3.5: Sample screens of the Contacts App

3.4 Regular Work

After a couple of weeks of learning and building my own projects, my senior programmer assigned my directly in their on going real world project. I will thrilled and scared at first as I had no prior experience building an app that would be used by thousands of users nationwide.

Every working day started off with attendance in Fountain IT & Software Services' online attendance application AttendR. A brief team meeting over zoom was held where each of us would give the update of their share of work, any challenges that needed consulting/ modification and finally, finished off with any upcoming/ pending work of that day. All projects that the company was working on was skilfully managed in an online project management app called Trello.

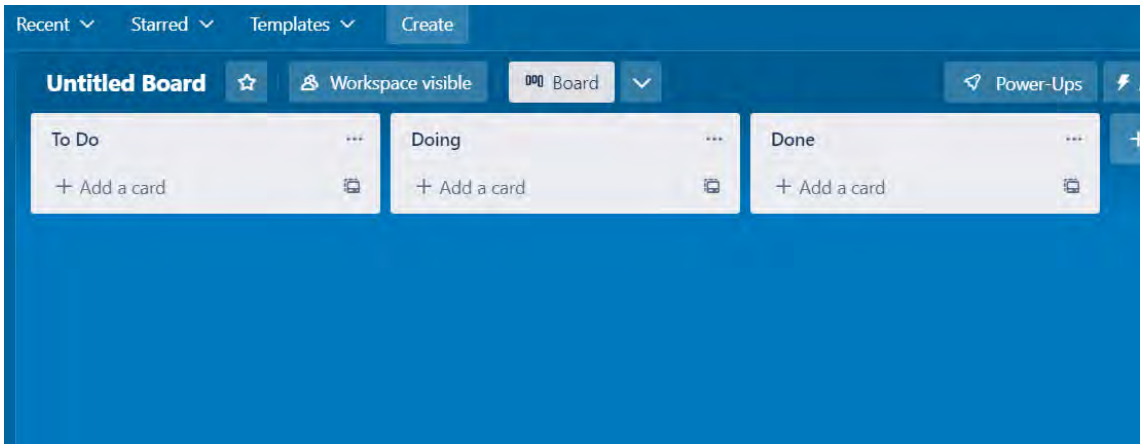


Fig. 3.6: Trello Desktop App snippet

I had to work on Android Studio for developming android apps and was assigned a branch called dev-tariq in Gitlab. All updates were pushed to my gitlab branch via GitBash or through Push Requests in Android Studio itself. Those Push Requests were then sent as Merge Requests in Gitlab where a supervisor would check its contents and finally merge my updates with the master branch.

The first few months went by designing layouts from Figma files in Android Studio. A sample design page from Android Studio is provided on the following page:

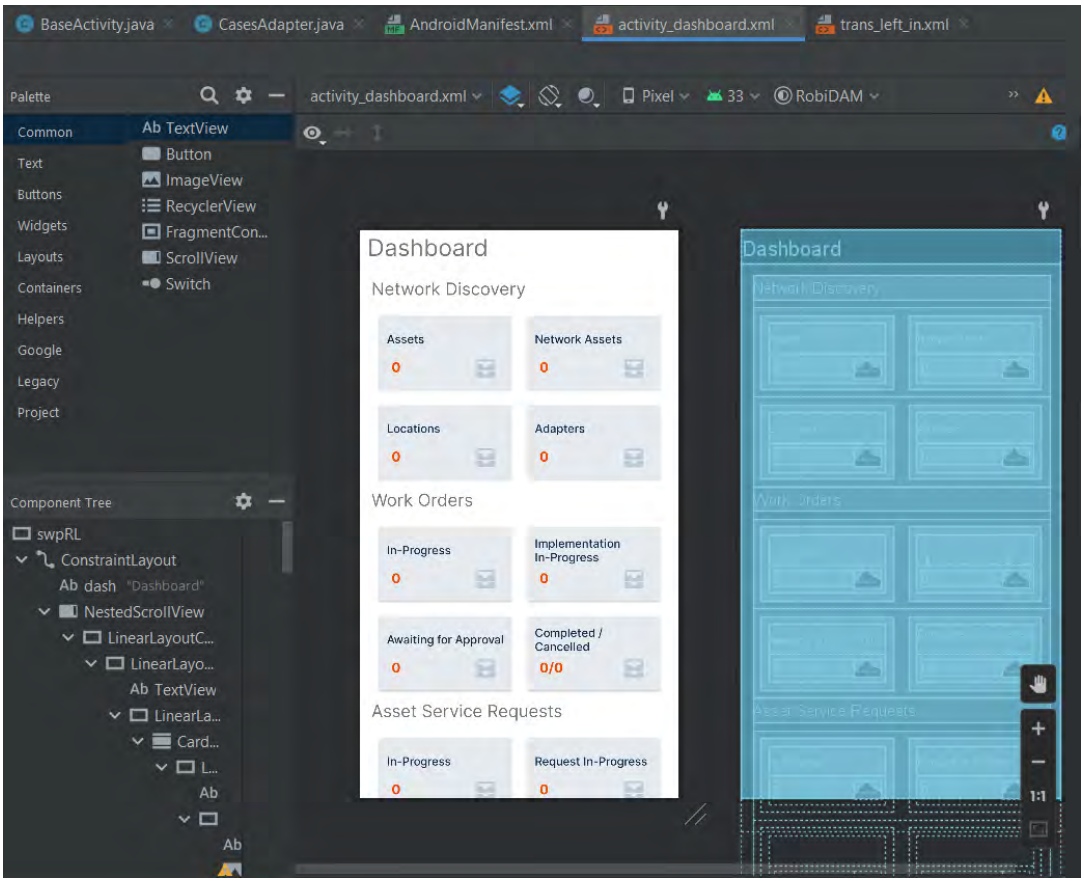


Fig. 3.7: Android Studio Dashboard UI

Although I used to drag and drop UI components from the palette in the beginning, I soon got strong hold of the xml coding syntaxes and standards and began coding them to further customize the look of the app.

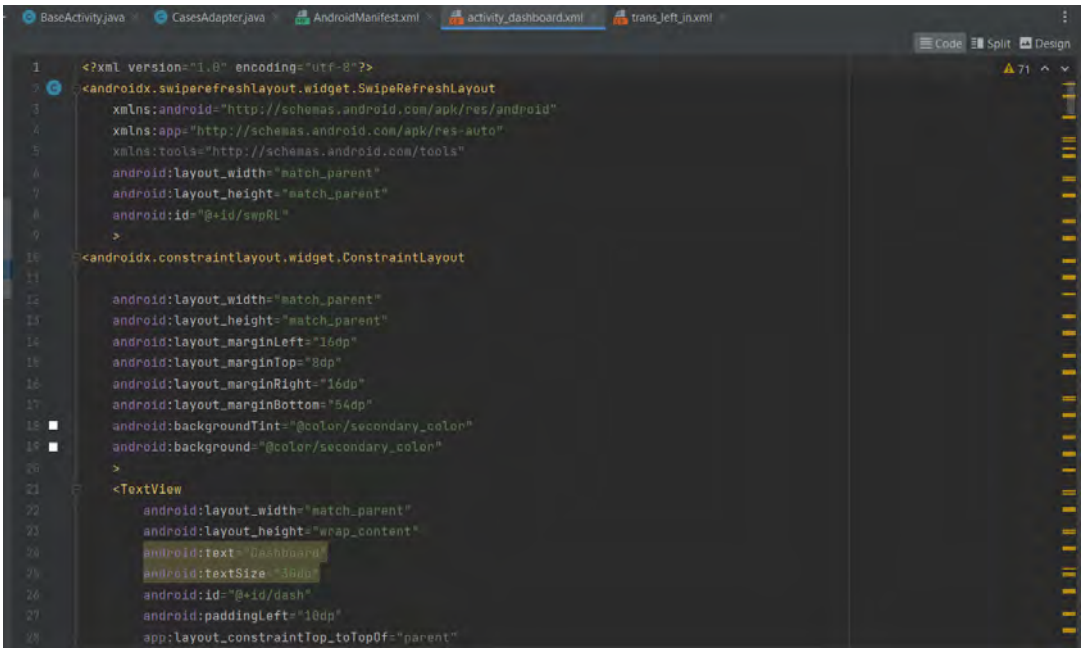


Fig. 3.8: Dashboard Design Code in xml

As the number of layouts increased, so did the complexity. There were over 100 UI layouts including fragments, navigationViews, custom TextViews, Sidebars, Menu items and toolbars.

This meant that every new screen had to be built from scratch. I knew it was impossible to finish all of them within the alloted project duration even with the help of my seniors. Google has aleady provided enough support materials and documentation to help design large, complex android applications. It was a necessity to study them to master fast paced design and UI implementation. And that is exactly what I did.

The programming part was my favourite. The UI elements needed to be logically connected to the backend program that was referenced in distinct java files. I used my pre existing java skills and had to learn a lot of new technologies to implement and integrate third party libraries aswell. After the initial design and coding was complete, most of my time was spent on debugging and fine tuning my code to optimize performance and scalability of my code. There were days where a single problem was addressed. My team, on the other hand, was busy preparing the necessary APIs and backend databases.

I was busy consulting them anytime a new feature needed to be implemented in the android app, esp. in the network and connectivity part of my app. Indeed, it was my supervisor that dealt with critical android components that was too difficult and beyond my scope at that time.

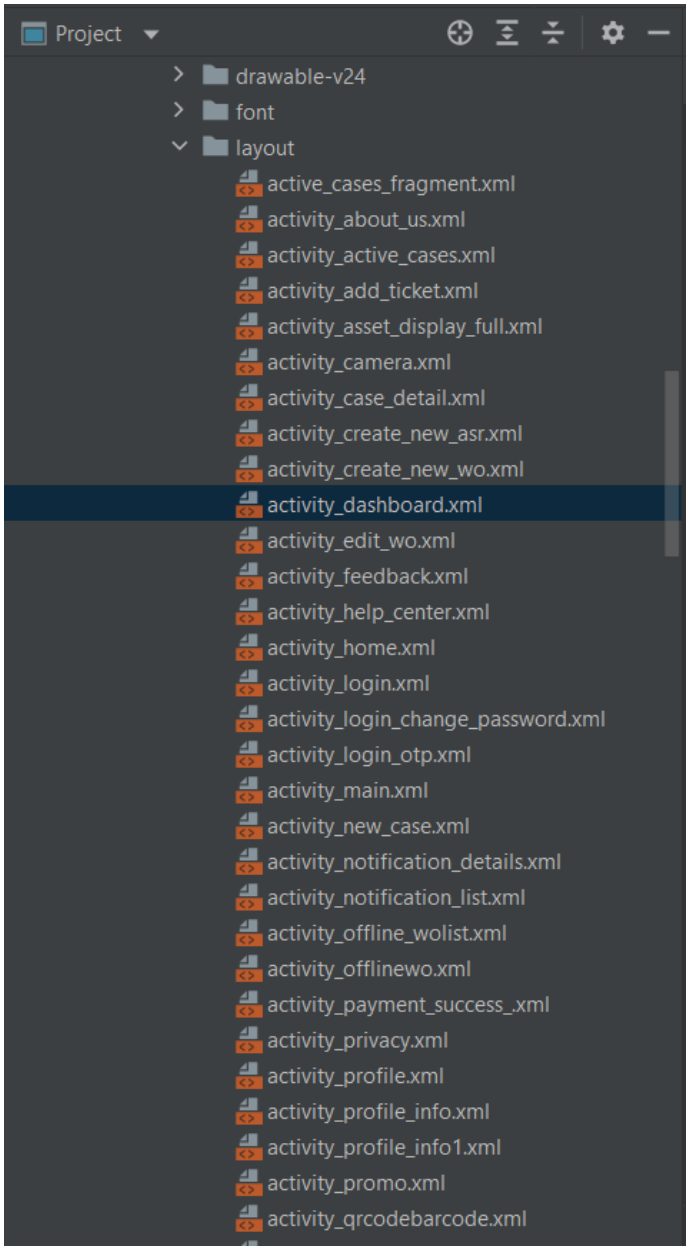


Fig. 3.9: Project's UI Layout Files

Chapter 4

Analysis

4.1 Design Palette

To start off, all Views (widgets, views, shapes, images, buttons, texts, toolbar, etc.) need to be placed inside atleast 1 layout (ViewGroup) which resides inside a .xml file:

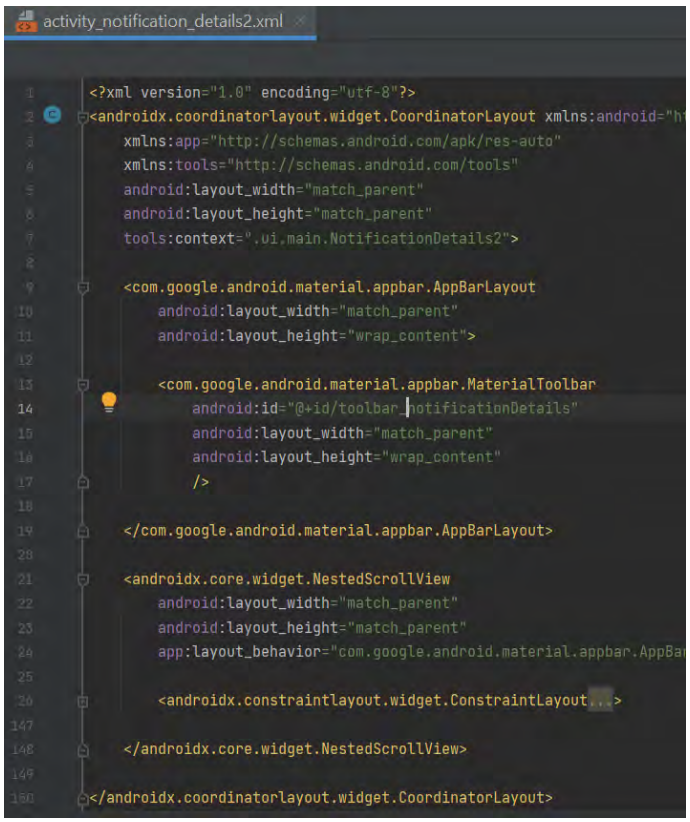


Fig. 4.1: UI Layout Design in xml format (nested hierarchy)

This is because android follows a hierarchy within the UI architecture. It is incumbent upon all design layout files to abide by this. Technically, these are all java classes with parent child relationship, same as inheritance. The .xml files just make it easier for developers to design and work on an app's UI/UX.

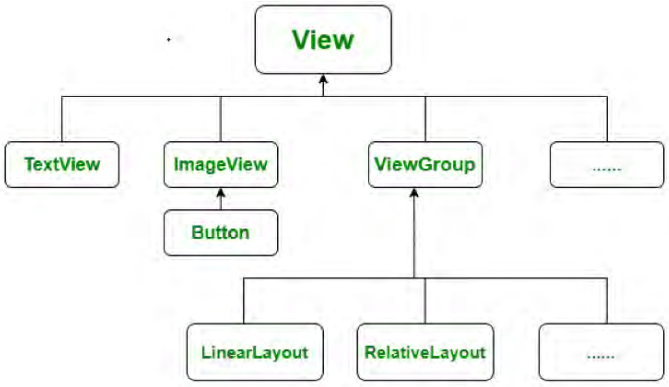


Fig. 4.2: View Hierarchy

However, if the design is difficult to code or complex, then it is convenient and recommended to take help from 3rd party applications. Android Studio is compatible with a variety of file types including .jpg, .png, .svg, etc. It is also recommended to import .svg files to add graphics and vectors:

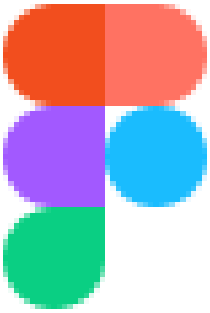


Fig. 4.3: Figma: To design layout elements

↓

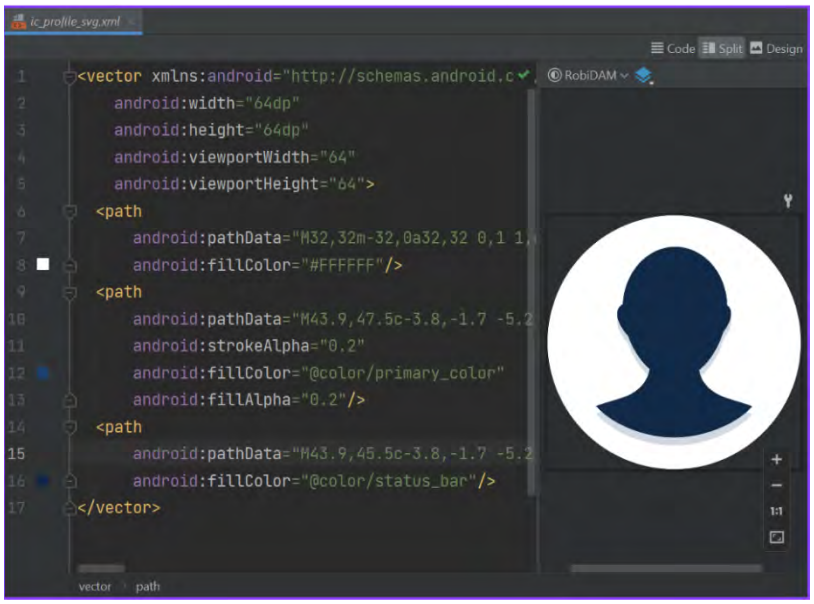


Fig. 4.4: Import Figma file as Drawable Resource File

↓

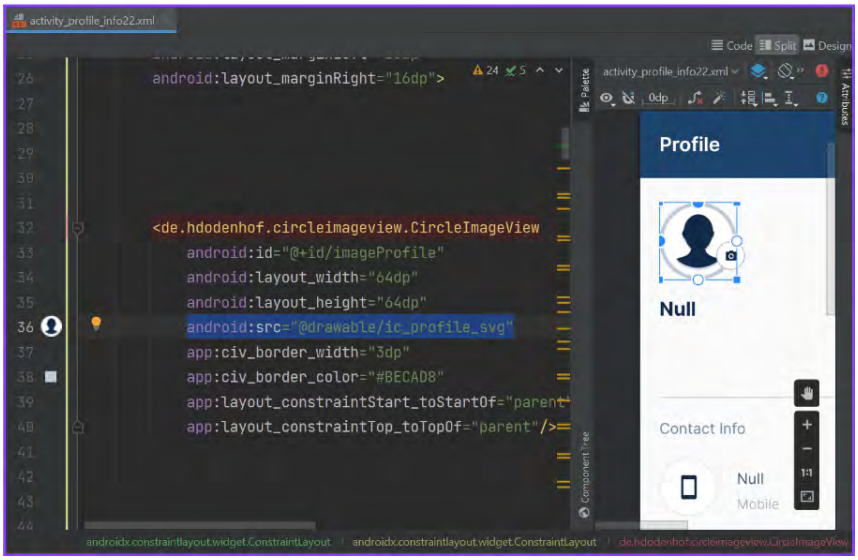


Fig. 4.5: Attributing source file by adding Drawable as source

Android Studio features a dedicated tab for all resources that are used in the project.

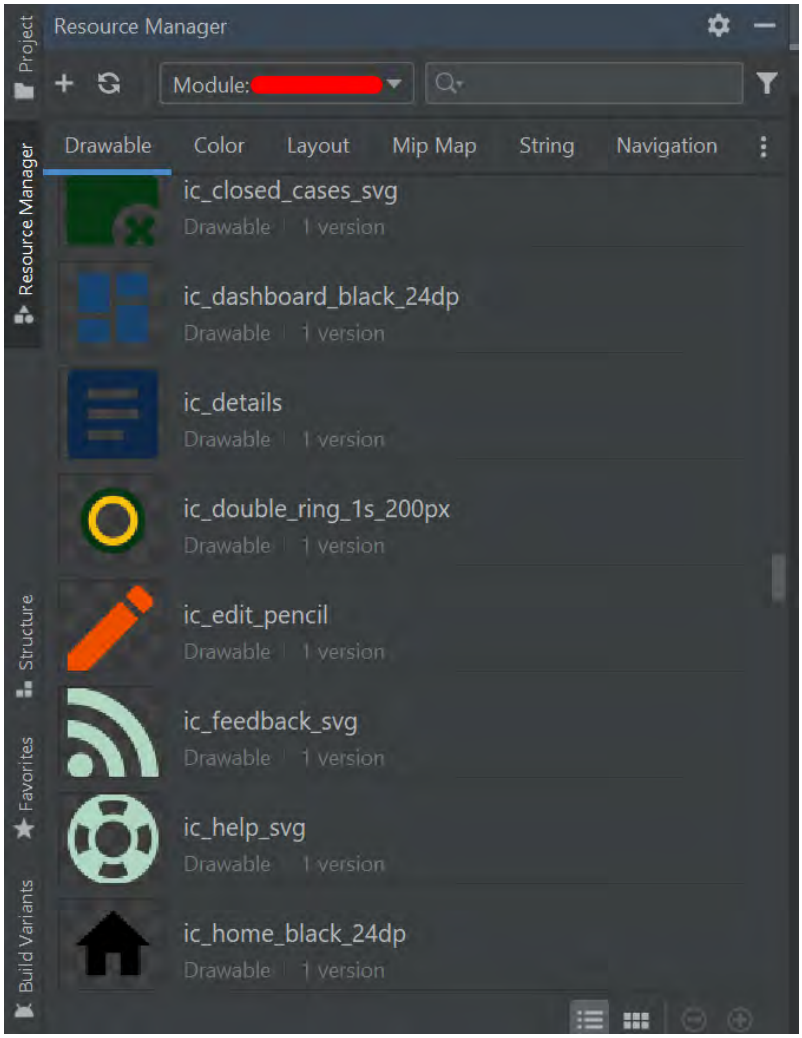
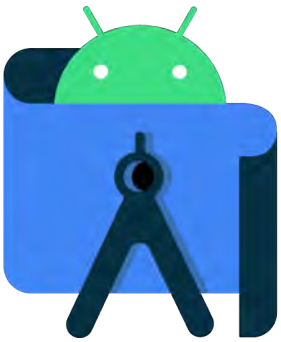


Fig. 4.6: Drawable tab under Resource Manager

Developers tend to have their own preferences when working on their field but almost unanimously, here are the dire necessities for any android app developer:



(a) Android Studio Logo

(b) Stack Overflow Logo

(c) Java Logo

Fig. 4.7: Every Android developers' essentials

Android studio is a framework designed by the android DevOps team of google to help developers built, debug and release android apps.

Stack Overflow is a forum website for professional and enthusiast programmers. It is the best place of the Stack Exchange Network consisting of thousands of active experienced developers helping out anyone in computer programming.

Java is the programming language that is used to program android apps. although google has developed its own language, Kotlin, for future app development, java is still widely used for modern app development.

4.2 Customising UI designs

Moving on, before the focus goes to designing UI within a layout resource file (*.xml* file), it is important to know that there is a precedence that is followed by android when it builds the app:

1. Applying character- or paragraph-level styling via text spans to TextView-derived classes
2. Applying attributes programmatically
3. Applying individual attributes directly to a View
4. Applying a style to a View
5. Default styling
6. Applying a theme to a collection of Views, an activity, or your entire app
7. Applying certain View-specific styling, such as setting a *TextAppearance* on a TextView

So, it is important to keep this in mind before applying a style in any UI element. Otherwise, some styles will not show up once the app is running as they will be overridden by android's default themes and styles' attributes.

Note that attributes can also be added programmatically. It is very useful to design this way in certain situations where small amount of code can result to quick style changes. Although there is very little evidence of any performance issues in doing so, it becomes mandatory to program styles in .java files in certain situations such as setting up the title of a toolbar with data which is fetched from an API endpoint:

```
private void initViewsAndValues() {
    toolbar=findViewById(R.id.toolbar_notificationDetails);
    setSupportActionBar(toolbar);
    getSupportActionBar().setHomeButtonEnabled(true);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    notificationInfo= (Notification) getIntent().getExtras().getSerializable( key: "notificationObject");
    try {
        if (notificationInfo.getData().getBody().toLowerCase().contains("work order")){
            getSupportActionBar().setTitle("WorkOrder " + notificationInfo.getData().getWorkorderId());
              toolbar.setTitle() and many more methods don't work for Support Toolbar (Appcompat v7) ...;
              this.setTitle(); also works
```

Fig. 4.8: Implementing toolbar (*SupportActionBar*) programmatically

There has been multiple deprecation of android libraries' packages and tools and new improvements and releases for more than the last 10 years. This calls for a necessity for a developer to stay up to date with latest releases and bug fixes and deprecations. All libraries have a range of Google's Android APIs (currently its API 32 for Android 12) within which it is functional. Use of deprecated libraries is a big risk as it will mostly cause app crashes and difficultly in maintaining an app post deployment. Therefore, it is essential to use the latest libraries for a maximum life of the app and constantly look for any packages of android that requires a stable and new substitution.

For starters, reusable shapes should be created in a Drawable Resource File to avoid redundancy in styling:

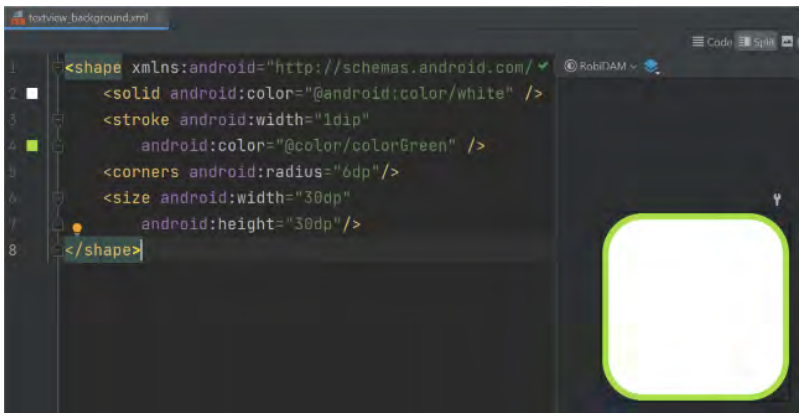


Fig. 4.9: Sample Drawable Resource File

Here are some of the important issues that need consideration during app design:

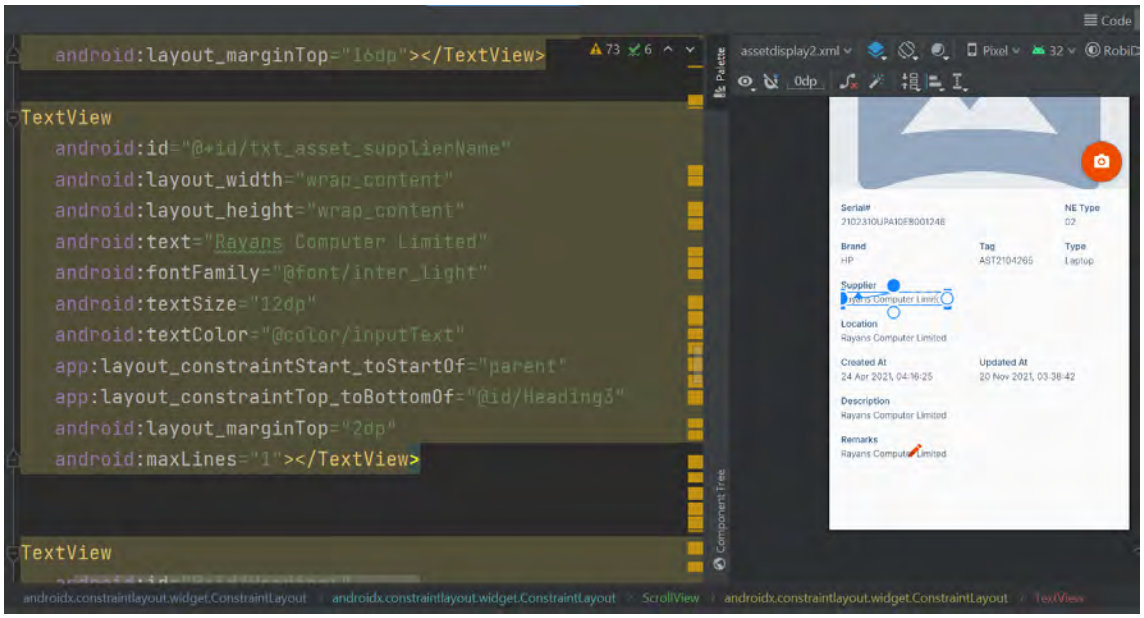


Fig. 4.10: Hardcoded style attributes

Note that the attributes of TextView:$android:id = "@+id/txt\_asset\_supplierName"$ are all hardcoded on the *.xml* file of *AssetDisplay.xml*. This will do the job of styling the text appearance of this text.

But this is bad practice as there are many other textViews with same attributes such as $android:fontFamily$, $android:textSize$, $android:textColor$ and $android:maxLines$. This is boilerplate code and this can be easily solved if the styles of all the textViews are inherited from a single ¡style¿ key.

```
<style name="myTextAppearance" parent="@android:style/TextAppearance">
    <item name="android:textSize">12dp</item>
    <item name="fontFamily">@font/inter_medium</item>
    <item name="android:textColor">@color/purple_200</item>
</style>
```

Fig. 4.11: Creating a new style named *myTextAppearance*

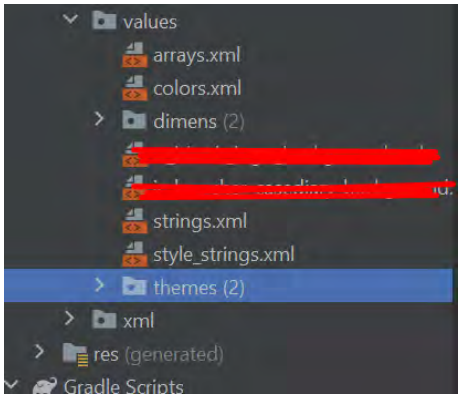


Fig. 4.12: Styles are placed under the *themes.xml* file. By default, *themes.xml* file consists of all the styles, built-in and custom, that other widgets such as TextViews can inherit from.

A proper implementation of such styling is illustrated below:

The TextView $android : id = "@ + id/wo\_description2"$ inherits its styles from a style named textHeading from the $< style >$ tag.

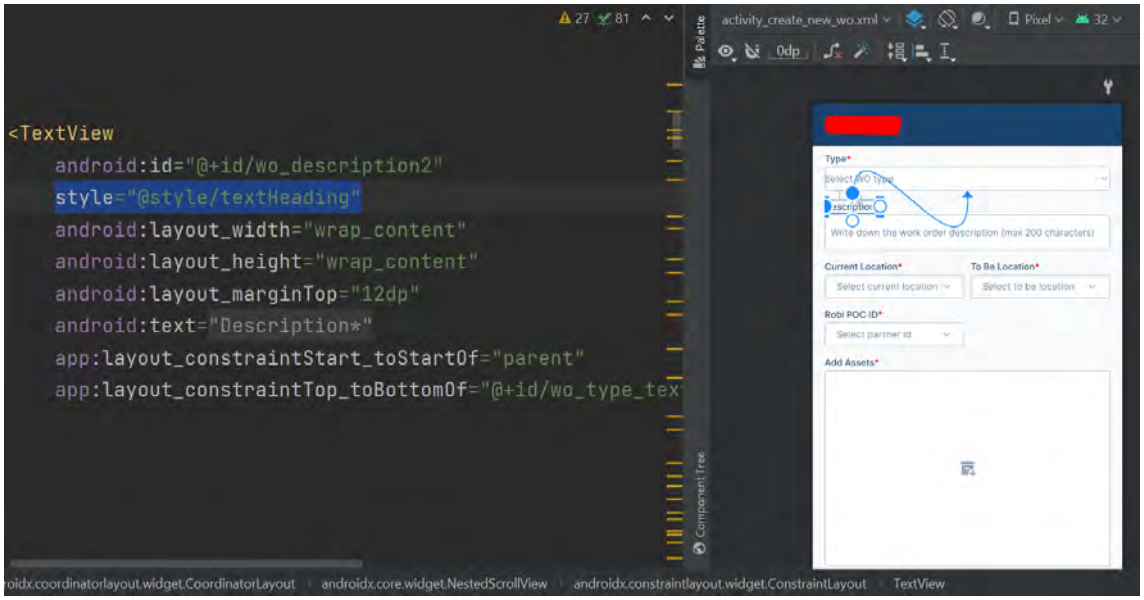


Fig. 4.13: Textview inheriting style from $@style/textHeading$ attribute

And the respective style named $textHeading$ is located in the $themes.xml file$. In this way, all the heading textViews are styled by a simple reference to styles in $theme.xml$ file. The color of 15 headings (typed in bold blue color) can be changed by changing $android : textColor$ value under $textHeading$ style attribute.

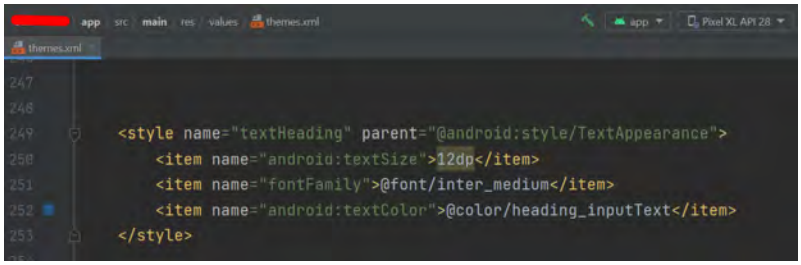


Fig. 4.14: Custom style $textHeading$ in $themes.xml$ file

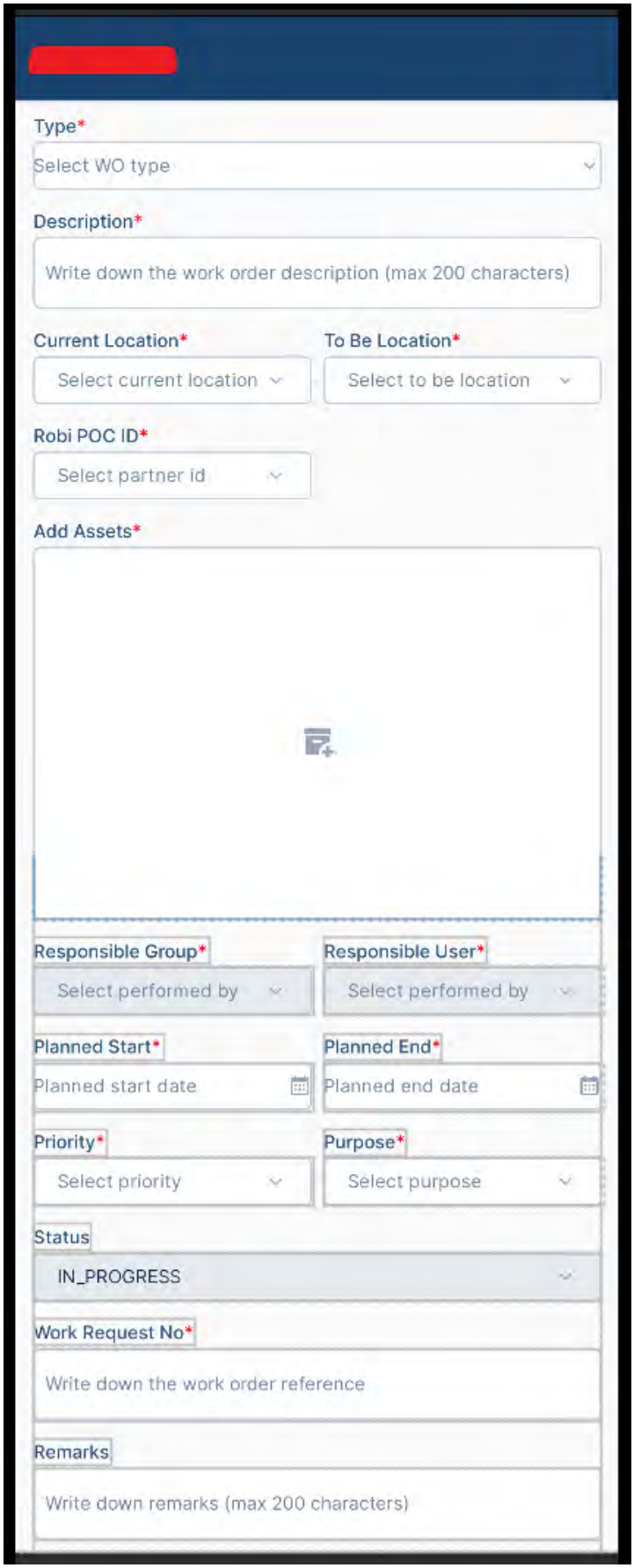


Fig. 4.15: Final look of a UI screen

4.3 Industry Standards

But if this is such a simple and easy task, so why does it take such time to implement it?

The answer is that there are many widgets, tools and views that are needed to make an android app's UI look attractive. But more importantly, each of these are children of a View class which need to be studied, their documentation well read and using accurate overriding of the xml attributes. For example, the attribute $android : textColor$ will have no impact when styling an $ImageView$ simply because an $ImageView$ consists of an image only and does not contain any text.

The style attribute has to be added to each of the elements inside any layout file for it to override android's default styling attributes. This can also be dismissed by creating/overriding Android's default theme attributes inside the theme.xml file. Android has over a hundred default tags and their hundreds of attribute values for its default theme for the app. A manipulation of this hierarchy is shown below:

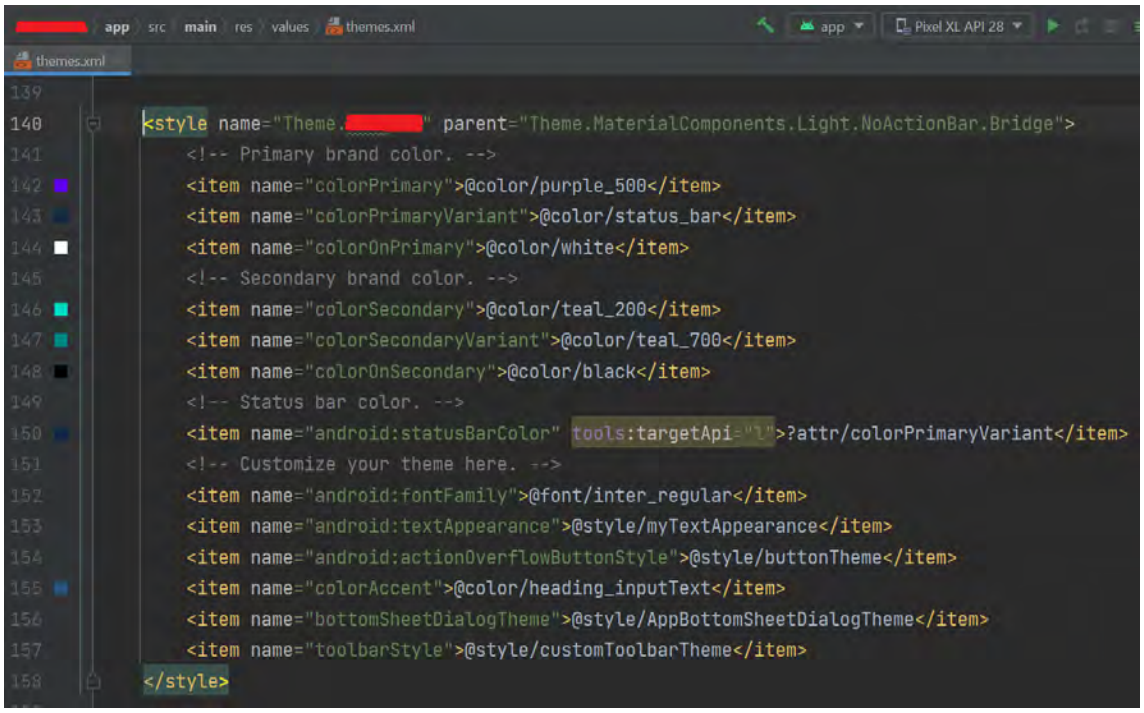


Fig. 4.16: Overriding theme's style attributes

Pressing Control+Q when mouse cursor is placed over our custom theme ¡style name="Theme.*" will reveal the default stylings as well as the overridden ones. An example is shown on the next page:

- It points to the toolbar theme overridden by $customToolbarTheme$.
- It points to the default themes that are provided in the android built in libraries.

Overriding theme does not need any style attribute to be coded in any layout file. The overridden themes automatically have effect on the views/widgets of the app.

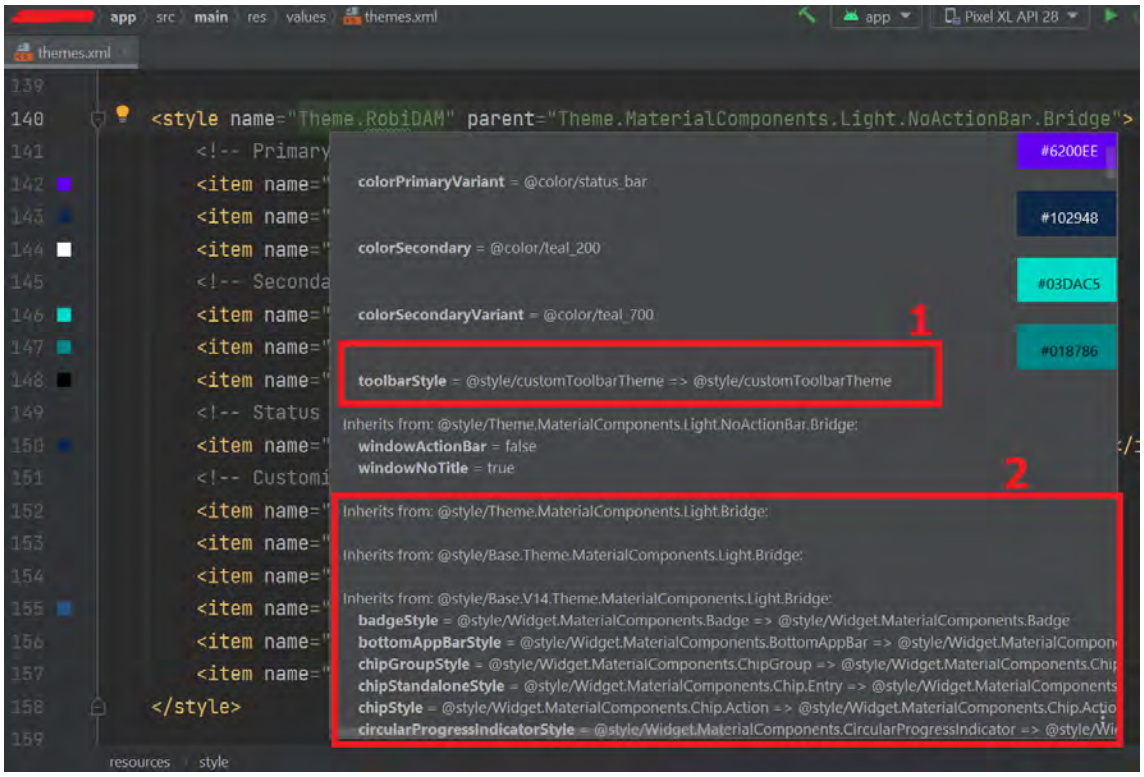


Fig. 4.17: Popup showing Library components

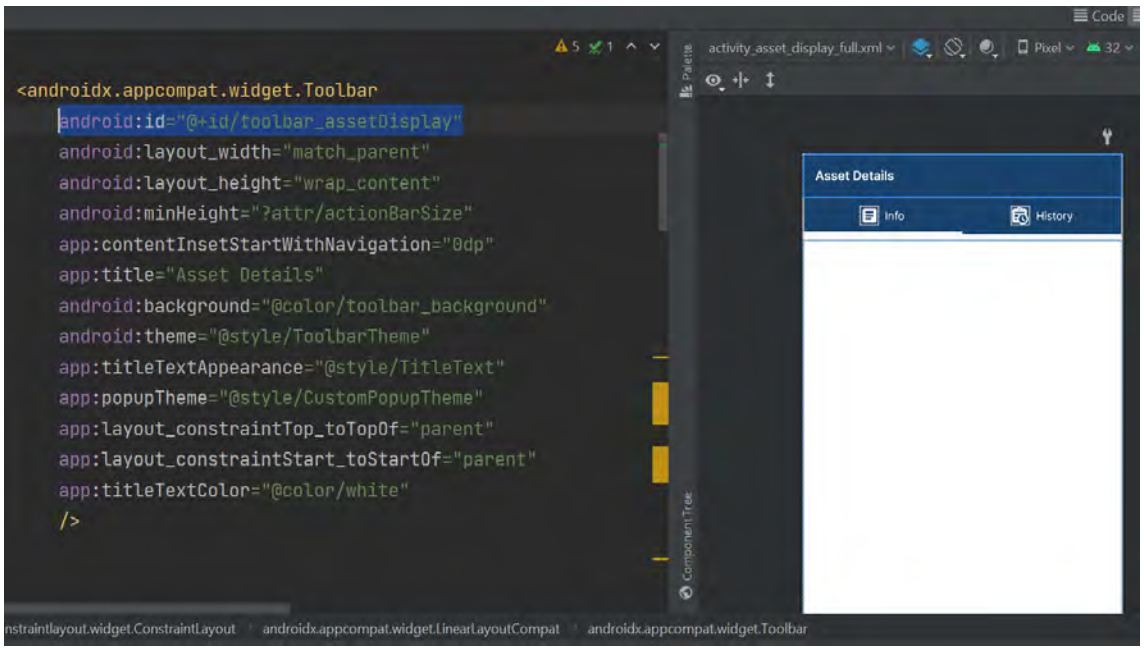


Fig. 4.18: Customizing Toolbar widget

Fig. 4.18 shows an inefficient implementation of toolbar styling (not recommended but does the job).

A excellent implementation of toolbar theme to automatically style all the toolbars of the entire application is illustrated as follows:

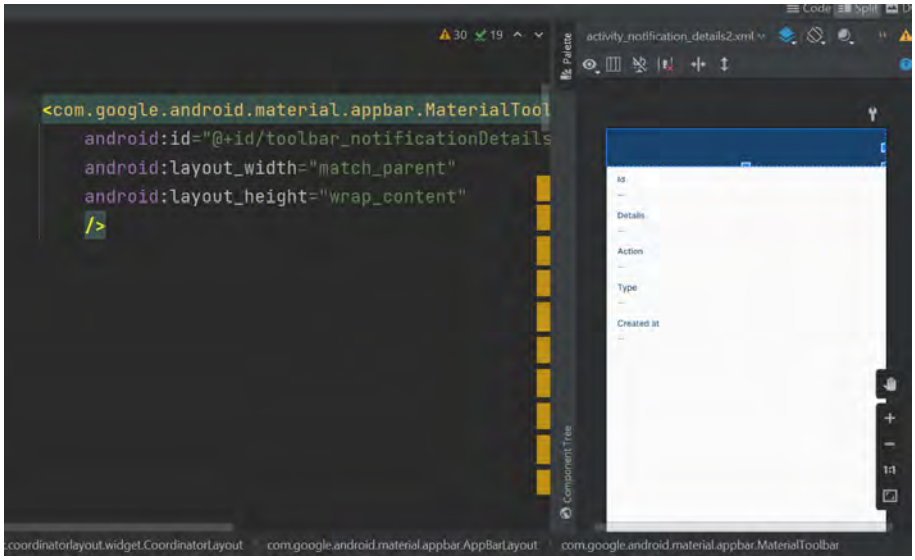


Fig. 4.19: Adding a generic toolbar

```
<style name="Theme.[redacted]" parent="Theme.MaterialComponents.Light.NoActionBar.Bridge">
    <!-- Primary brand color. -->
    <item name="colorPrimary">@color/purple_500</item>
    <item name="colorPrimaryVariant">@color/status_bar</item>
    <item name="colorOnPrimary">@color/white</item>
    <!-- Secondary brand color. -->
    <item name="colorSecondary">@color/teal_200</item>
    <item name="colorSecondaryVariant">@color/teal_700</item>
    <item name="colorOnSecondary">@color/black</item>
    <item name="toolbarStyle">@style/customToolbarTheme</item>
</style>
```

Fig. 4.20: Overriding *toolbarStyle* inside Theme's style attribute in *themes.xml*

```
themes.xml
156
157
158     <style name="customToolbarTheme" parent="@style/Widget.MaterialComponents.Toolbar">
159         <item name="titleTextAppearance">@style/TitleText</item>
160         <item name="titleTextColor">@color/white</item>
161         <item name="subtitleTextAppearance">@style/SubtitleText</item>
162         <item name="subtitleTextColor">@color/toolbar_subtitle</item>
163         <item name="android:background">@color/toolbar_background</item>
164         <item name="colorControlNormal">@color/white</item>
165         <item name="android:gravity">center_vertical</item>
166         <item name="popupTheme">@style/CustomPopupTheme</item>
167         <item name="contentInsetStartWithNavigation">0dp</item>
168         <item name="navigationIconTint">@color/white</item>
169     </style>
```

Fig. 4.21: Referencing the built *customToolbarTheme* Theme attributes in a separate style tag.

4.4 Functionality

The functionalities of UI elements are a developers next concern. A sample use case is built on the following example. The network integration was firstly tried in Postman, a software for checking performance and usability of APIs.

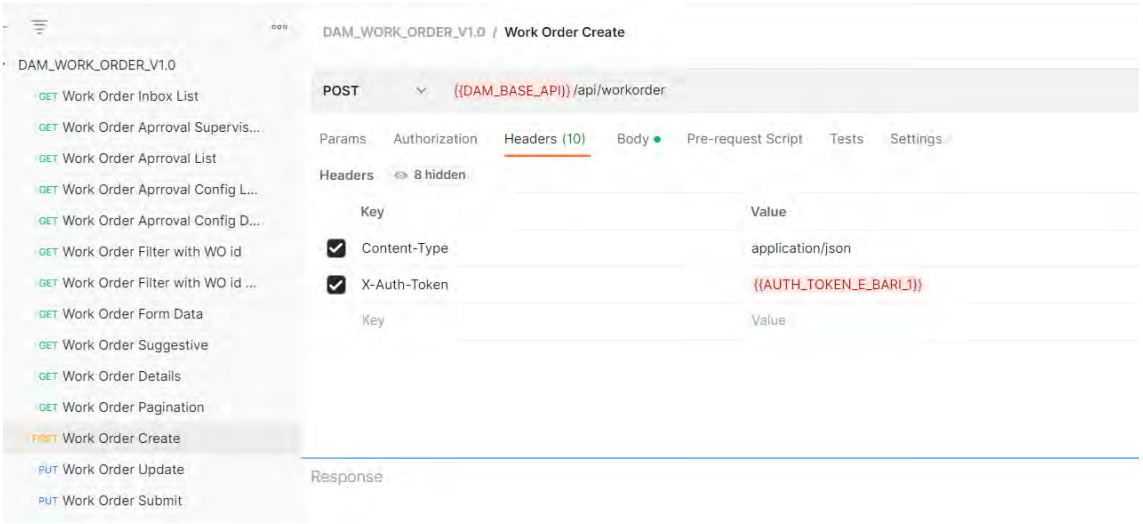


Fig. 4.22: Postman User Interface (API Testing)

A Java interface, such as the one in Fig. 4.23, required hundreds of lines of code where it was implemented. The verified APIs were coded in java as shown on the next page:

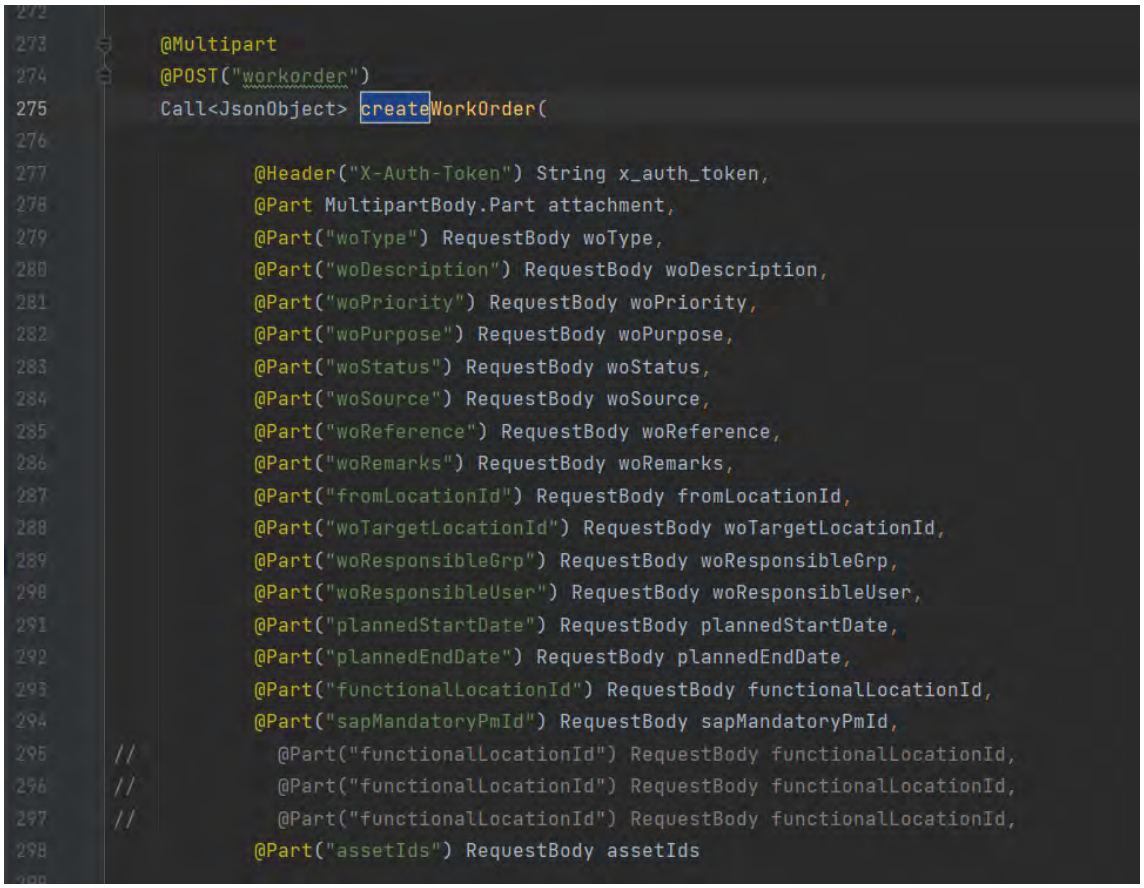


Fig. 4.23: Java Interface showing method parameters

After each module of the app was designed, coded and finalized, it was forwarded to the testers of my team to identify any issues of the app that needed any fixing/ modification. Each of the issues were further identified, fixed, revised and sent out again for testing. Some of it took days to reconcile. I got amazed how my supervisor would solve an excruciating bug in my code in a matter of seconds. I knew I was in good hands and never thought that learning could be so much fun and entertaining.

Chapter 5

Conclusion

Android development is very interesting and anyone can partake good developing knowledge from it. The designing issues need to be handled early in the app's production process so that size and complexity does not create havoc before deployment. It is a must for any and all developers and programmers to study the documentation and support materials to maximize their impact on the application's scalability, size and performance. The project that I worked on was a good hit! The clients loved the work and my seniors congratulated me on every milestone that I completed on my way. They were so pleased that they offered me to join FountainIT as a full-time permanent employee which I gleefully accepted. It was great learning experience that allowed me to connect to many senior and professional developers in IT. I would urge students to go for an internship in a good software firm where they can make the best of the time that the university has allotted for them. I thank BRAC university for allowing its students to undergo an internship as a part of the completion of their studies.

Bibliography

[1] *Android Mobile App Developer Tools - Android Developers*. URL: https://developer.android.com (visited on 01/01/2023).

[2] *GeeksforGeeks | A computer science portal for geeks*. URL: https://geeksforgeeks.org (visited on 01/01/2023).

[3] *Google for Developers - from AI and Cloud, to Mobile and Web*. URL: https://developer.android.com (visited on 01/01/2023).

[4] *Material Design*. URL: https://m3.material.io (visited on 01/01/2023).

[5] *Stack Overflow - Where Developers Learn, Share, & Build Careers*. URL: https://stackoverflow.com (visited on 01/01/2023).