

PRIVLOK

Submitted in partial fulfilment of the Requirements for the Degree of

Master of Computer Applications

A PROJECT REPORT

Submitted by

Sachin Rathi

(University Roll No 1900290140029)

Batch:2019-2022

Under the Supervision of

Dr. Vipin Kumar

(Associate Professor)



DEPARTMENT OF COMPUTER APPLICATIONS

DR.APJ ABDUL KALAM TECHNICAL UNIVERSITY

LUCKNOW

(Formerly Uttar Pradesh Technical University, Lucknow)

DECLARATION

I hereby declare that the work presented in this report Privlok was carried out by our team. I have given this credit to our Respected Vipin Sir for all the ideas, diagrams, graphics, computer programs, experiments, results. I have used quotation marks to identify sentences and given credit to the original sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results. I shall be fully responsible and answerable.

Name : Sachin Rathi

Roll. No. : 1900290140029

Branch : Master of Computer Applications

CERTIFICATE

Certified that Sachin rathi(1900290140029) has carried out the project work presented in this report entitled “Privlok” for the award of Master of Computer Application from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

Dr. Vipin Kumar

Professor

Dept. of Computer Applications KIET Group of Institutions, Ghaziabad

Date:

ABSTRACT

This chapter guides user through Android application development in Android Studio. It starts by creating an Android application project for phones and tablets and continues with additional development modules for the application. Creating a new Android project is pretty straightforward with Android Studio. Android Studio helps to select the best SDK version for the application. One needs to build the application and launch it on a device after creation of the application project. If a user already has a device attached to the development machine with a compatible Android SDK version, he/she can create a virtual Android device to run the application. The chapter also covers the basic building blocks of Android applications and the capabilities of Android Studio. The main building blocks of Android applications are activities, services, assets, XML files, the Android Manifest file, and modules.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my Project supervisor, Dr. vipin Kumar, Professor Department of Computer Applications for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Sachin Rathi

Table of Contents

DECLARATION	2
CERTIFICATE	3
ABSTRACT	4
ACKNOWLEDGEMENT	5
CHAPTER 1	8
INTRODUCTION	8
1.1 PROJECT DESCRIPTION	8
1.2 ANDROID AND ITS CHARACTERISTICS	9
	11
1.3 ANDROID ANALYSIS USING SQLITE	12
• Text Messages/MMS	12
• Browser History	12
• Social Networking	13
WhatsApp	13
Facebook	13
Skype	14
CHAPTER 2	17
LITERATURE REVIEW	17
2.1 ANDROID	17
2.2 SQLITE	18
Database – Package	18
Database – Creation	18
Database – Insertion	19
CHAPTER 3	20
REQUIREMENTS	20
3.1 OBJECTIVE	20
3.2 IMPORTANCE	21
CHAPTER 4	22
PROJECT WORK FLOW	22
Android Studio for beginners, Part 1: Installation and setup	22
Get started with Android Studio	23
Download Android Studio	24
Windows requirements	24

Mac OS requirements	24
Linux OS requirements	25
Android SDK command-line tools	26
Installing Android Studio on 64-bit Windows 10	26
Running Android Studio	33
Reconfigured buttons	49
Accessing AVD Manager and SDK Manager	52
The Project and editor windows	52
• Branch names and directory/file names	54
CHAPTER 5	55
CODING	55
5.1 INTERMEDIATE CODE	55
privlok/Main.activity.java	55
privlok/add activity	56
privlok/add activity.xml	58
privlok/bottom navigation activity	61
privlok/mainactivity.xml	62
5.2 UPDATE CODE	63
privlok/(UPDATE PROJECT CODE) AddPassword.java	63
privlok/BottomNavigationActivity.java(update)	66
privlok/CustomAdaptar.java(update)	67
privlok/bottomNavigationActivity.xml(update)	69
privlok/datahelper.java(update)	71
privlok/listview.xml(update)	73
privlok/updateActivity.java(update)	74
privlok/updateActivity.xml(update)	76
CHAPTER 6	79
APPLICATION FORMAT	79
6.1 STEP 1	79
6.2 STEP 2	80
6.3 STEP 3	81
6.4 STEP 4	82
6.5 STEP 5	83
6.6 STEP 6	84

CHAPTER 7	85
CONCLUSION	85
REFERENCES	86

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

Our project is based on android technology whose name is privlok. We are just made this application to store passwords of different different platform. The technology which we are using in our application are as follows .i.e. Android studio and SQL. And the benefits of our application in real world is that user no need to remember all the passwords all the time. User can access our application to retrieve all the passwords which have been stored.

1.2 ANDROID AND ITS CHARACTERISTICS

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance. The source code for Android is available under free and open source software licenses.

Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below –

Android OS basic screen provides a beautiful and intuitive user interface. GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

SQLite, a lightweight relational database, is used for data storage purposes. H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.

User can jump from one task to another and same time various application can run simultaneously. Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore.

The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence. The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.

 <p>Android 1.6</p> <p>Donut</p>	 <p>Android 2.0</p> <p>Eclair</p>	 <p>Android 2.2</p> <p>Froyo</p>	 <p>Android 2.3</p> <p>Gingerbread</p>	 <p>Android 3.0</p> <p>Honeycomb</p>
 <p>Android 4.0</p> <p>Ice Cream Sandwich</p>	 <p>Android 4.1</p> <p>Jelly Bean</p>	 <p>Android 4.4</p> <p>KitKat</p>	 <p>Android 5.0</p> <p>Lollipop</p>	 <p>Android 6.0</p> <p>Marshmallow</p>

1.3 ANDROID ANALYSIS USING SQLITE

Android supports application-specific relational databases that use SQLite for storing its data. Since SQLite databases are both lightweight and based on files, they, make an ideal platform for embedded devices. The default file format for SQLite databases is .db. The other file extensions, which denote the SQLite database, are .sqlite, .sqlite3, .sqlitedb, and .db3, depending on the versions of SQLite that has been deployed. During Android SQLite database analysis, the first step is to extract these SQLite database files from the concerned device. The locations of some of the most important applications in an Android device are:

- **Text Messages/MMS**

The text messages prove to be helpful for the forensicator as they show the entire conversations that have been done between the suspect and other. The text messages and the MMS are stored in the SQLite file named as mmssms.db

The location of this file is /data/data/com.android.providers.telephony/databases.

- **Browser History**

Apart from the default browser of Android, a user may also use different browsers like Google Chrome, Firefox, etc. The history of all the browsers is stored in SQLite .db file. The default Android browser stores its data in db file at the location

/data/data/com.android.browser.

History of other browsers such as Firefox is stored in files, as shown below:

places.sqlite, key3.db, search.db, formhistory.db, cookies.sqlite, cert8.db.

• Social Networking

In Android SQLite database analysis, the social networking applications such as Skype, WhatsApp, Facebook, etc. may prove to be helpful in revealing sensitive data that can turn out to be helpful for the investigators.

WhatsApp

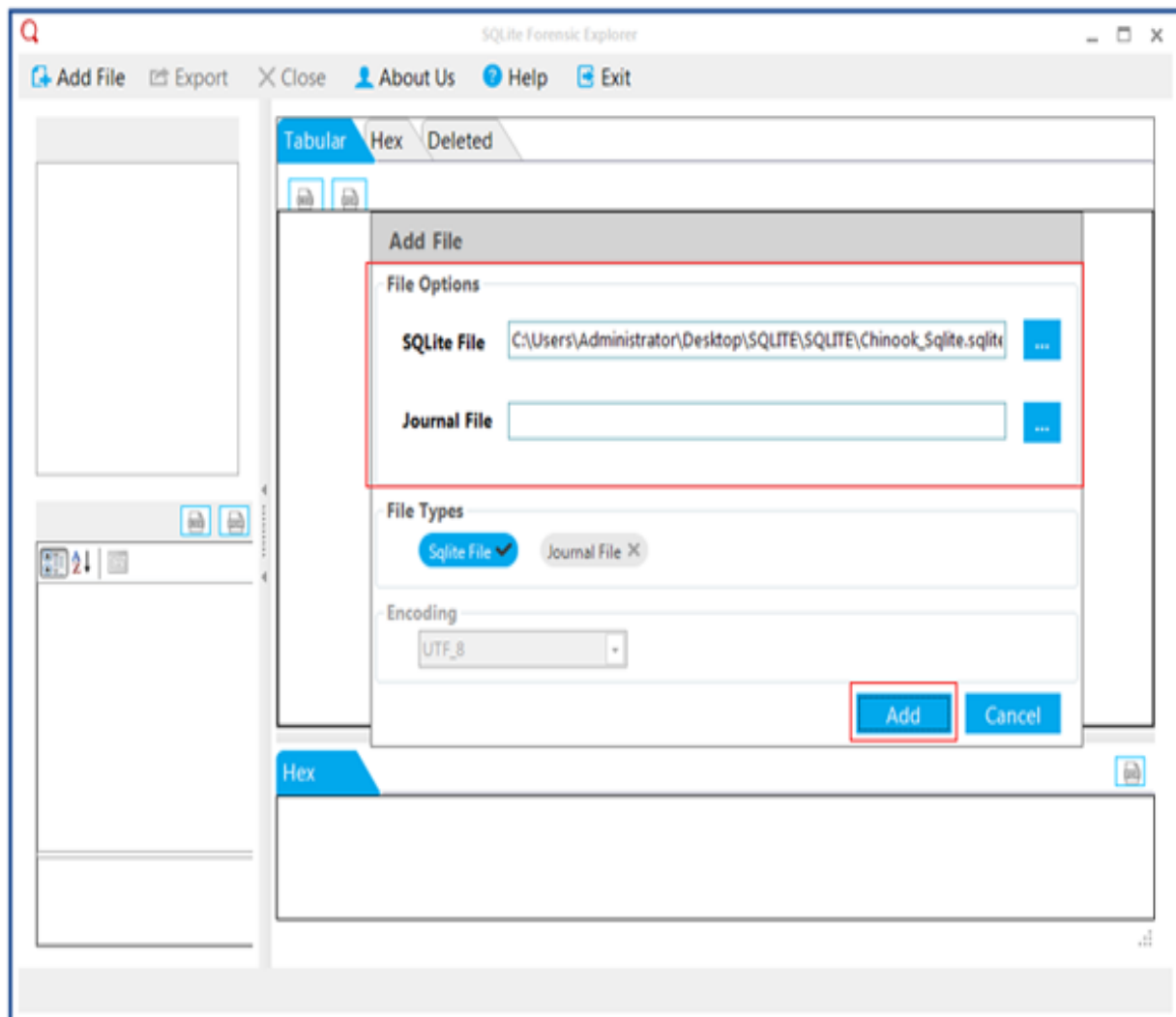
The data is stored in 2 SQLite files-msgstore.db.crypt8 stored at location /data/data/com.whatsapp/files/key locationwa.db file at the location /data/data/com.whatsapp/databases/wa.db.

Facebook

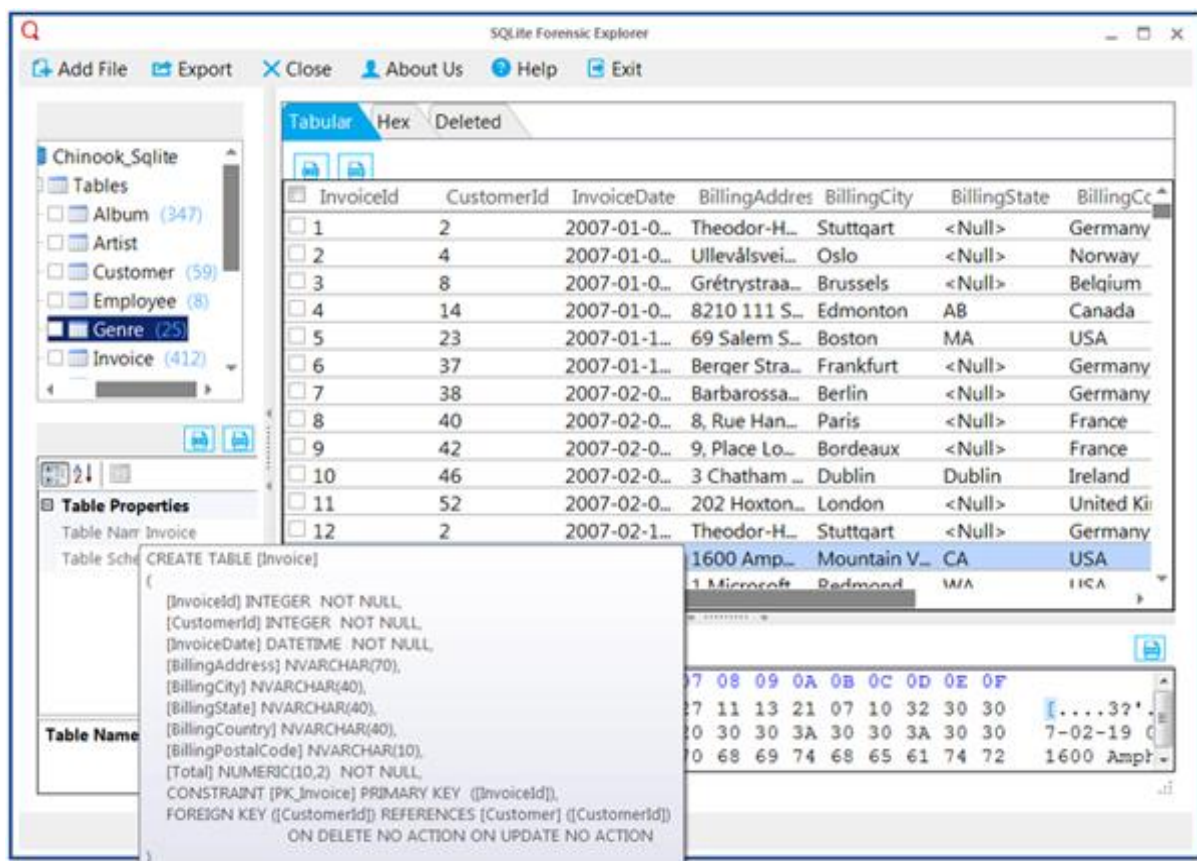
The entire data is stored in fb.db file at the location data/data/com.facebook.katana

Skype

Calls, contacts, messages, etc. are stored in main.db file the location /data/data/



Analyzing the SQLite database embedded in Android is the task that cannot be accomplished without the assistance of forensic tools. This is due to the reason that extraction and analyzing SQLite database is a very complicated procedure if done manually. Therefore, forensic utilities such as SQLite Database Viewer enable the forensicators to perform a deep analysis of SQLite data of Android device. The software does not impose any limitation on the size of the SQLite database file that needs to be explored. It supports recovery of deleted as well as secured items from the Android SQLite database.



The additional benefits that the software offers are:

- Gives the provision of indexing the SQLite database of Android.
- Enables to recover evidence from corrupted Android SQLite database.
- Color schemas to differentiate the data types like deleted, normal, and secured data.
- Support for analyzing BLOB data type within Android database.

CHAPTER 2

LITERATURE REVIEW

2.1 ANDROID

Android is a software package and linux based operating system for mobile devices such as tablet computers and smartphones. It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used. The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

There are many code names of android such as :

Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc.

Android is an open source and Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. This tutorial will teach you basic Android programming and will also take you through some advance concepts related to Android application development.

2.2 SQLITE

SQLite is a open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c.

Database – Package

The main package is android.database.sqlite that contains the classes to manage your own databases.

Database – Creation

In order to create a database you just need to call this method `openOrCreateDatabase` with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.

Its syntax is given below

```
SQLiteDatabase mydatabase = openOrCreateDatabase ("your database name",MODE_PRIVATE,null);
```

Database – Insertion

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class.

Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint (Username  
VARCHAR,Password VARCHAR);"); mydatabase.execSQL("INSERT INTO TutorialsPoint  
VALUES('admin','admin');");
```

CHAPTER 3

REQUIREMENTS

3.1 OBJECTIVE

All data stored in a structure manner

privacy

totally free

easy to use

better security

3.2 IMPORTANCE

It stores all your password , login information and records safely in one secure location that is protected with a password of your choice.

Save all sensitive data in a structure manner.

User remember their password all time.

No worry about loosing their data.

CHAPTER 4

PROJECT WORK FLOW

Android Studio for beginners, Part 1: Installation and setup

For some years now it's been clear that [Android](#) is a force to be reckoned with in the mobile OS landscape. This Java-based technology has sparked a new [gold rush](#), with programmers competing to [make money from their mobile apps](#). Android jobs are also plentiful, as shown by a quick job search using [Indeed.com](#).

To be successful, Android developers need a good grasp of the [Java language](#) (or [Kotlin](#)), [Android APIs](#), and [Android application architecture](#). It's also important to use an appropriate and effective development environment. For many years, [Eclipse IDE with the ADT plugin](#) was the preferred platform for Android development. Today, it's [Android Studio](#).

[[Also on InfoWorld: 25 simple tools for building mobile apps fast](#)]

What's new in Android Studio 3.x

Find out [what to look for in the latest version of Android Studio](#), including support for Kotlin, Java 8, and a wealth of new tools and plugins.

If you're new to Android Studio, this tutorial series will get you started. I'll briefly introduce the Android development platform, then show you how to download, install, and run the software. After that, we'll spend most of our time actually using Android Studio to develop an animated mobile app:

- In **Part 1**, you'll start up your first Android project and get to know Android Studio's main window.
- In **Part 2**, you'll **code the app**, learning how to use Android Studio to enter source code and resources into the project.
- In **Part 3**, we'll **build and run the app**, using both an emulated hardware device and a Kindle Fire tablet.
- In **Part 4**, I'll show you how to use built-in tools and plugins to **debug Android and improve your coding productivity**.

Examples in this series are from the most stable version of Android at the time of this writing, Android 3.2.1.

Get started with Android Studio

Android Studio is Google's officially supported IDE for developing Android apps. This IDE is based on **IntelliJ IDEA**, which offers a powerful code editor and developer tools. Android Studio 3.2.1 includes the following features:

- 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 help you 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 and [Google App Engine](#)
- Plugin architecture for extending Android Studio via plugins.

Download Android Studio

Google provides Android Studio for the Windows, Mac OS X, and Linux platforms. You can [download Android Studio](#) from the Android Studio homepage, where you'll also find the traditional SDKs with Android Studio's command-line tools. Before downloading Android Studio, make sure your platform meets the following requirements:

Windows requirements

- Microsoft Windows 7/8/10 (32-bit or 64-bit)
- 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Mac OS requirements

- Mac OS X 10.10 (Yosemite) or higher, up to 10.13 (High Sierra)

- 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Linux OS requirements

- GNOME or KDE desktop. Tested on Ubuntu 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications)
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
- 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Once you've ensured that your operating system is compatible with [Android Studio 3.2.1](#) or higher, download the appropriate Android Studio distribution file. The Android Studio download page auto-detected that I'm running a 64-bit Windows operating system and selected `android-studio-ide-181.5056338-windows.exe` (927 MB) for me to download.

Android SDK command-line tools

`android-studio-ide-181.5056338-windows.exe` includes an installer and the Android SDK command-line tools. If you don't need or want to use Android Studio, you can [download only the Android SDK command-line tools](#).

Installing Android Studio on 64-bit Windows 10

I launched `android-studio-ide-181.5056338-windows.exe` to start the installation process. The installer responded by presenting the **Android Studio Setup** dialog box shown in Figure 1.



Figure 1. Set up Android Studio

Clicking **Next** took me to the following panel, which provides the option to decline installing an Android Virtual Device (AVD).

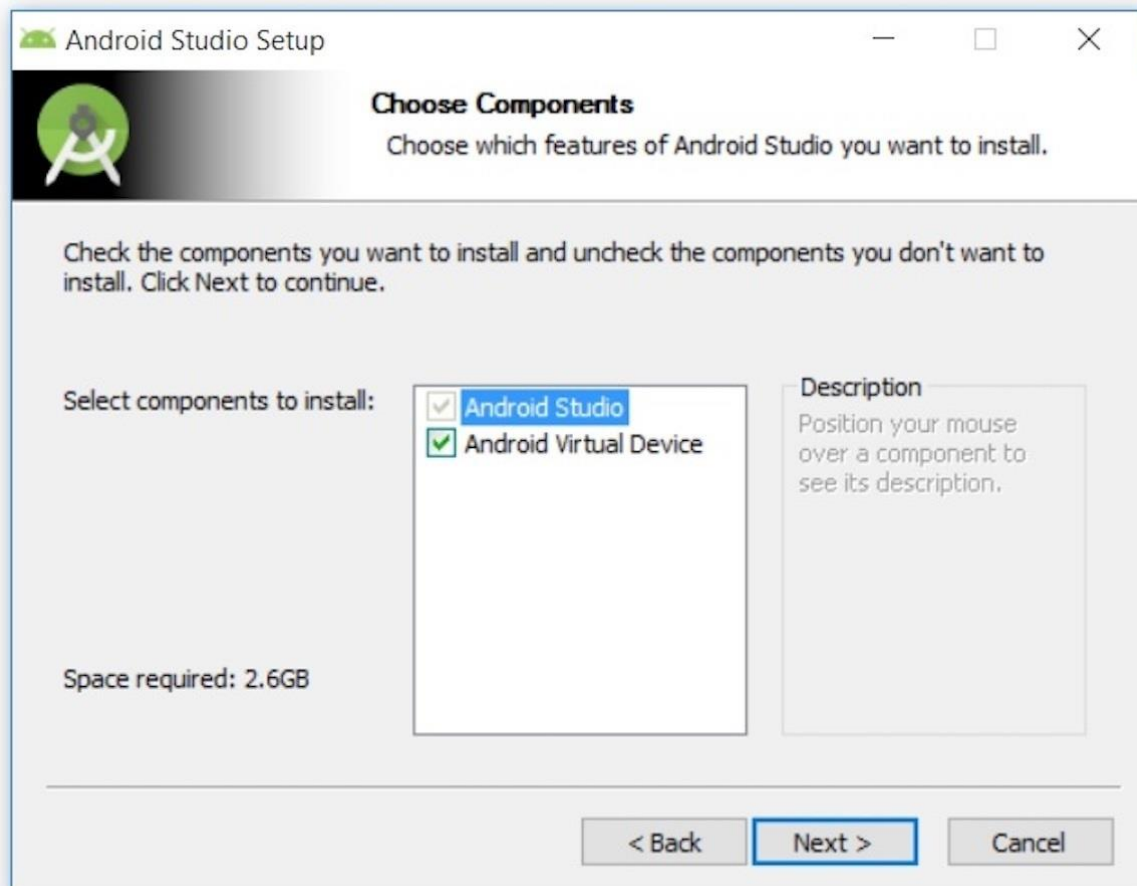


Figure 2. Install an Android AVD?

I chose to keep the default settings. After clicking **Next**, I was taken to the **Configuration Settings** panel, where I was asked to choose where to install Android Studio.

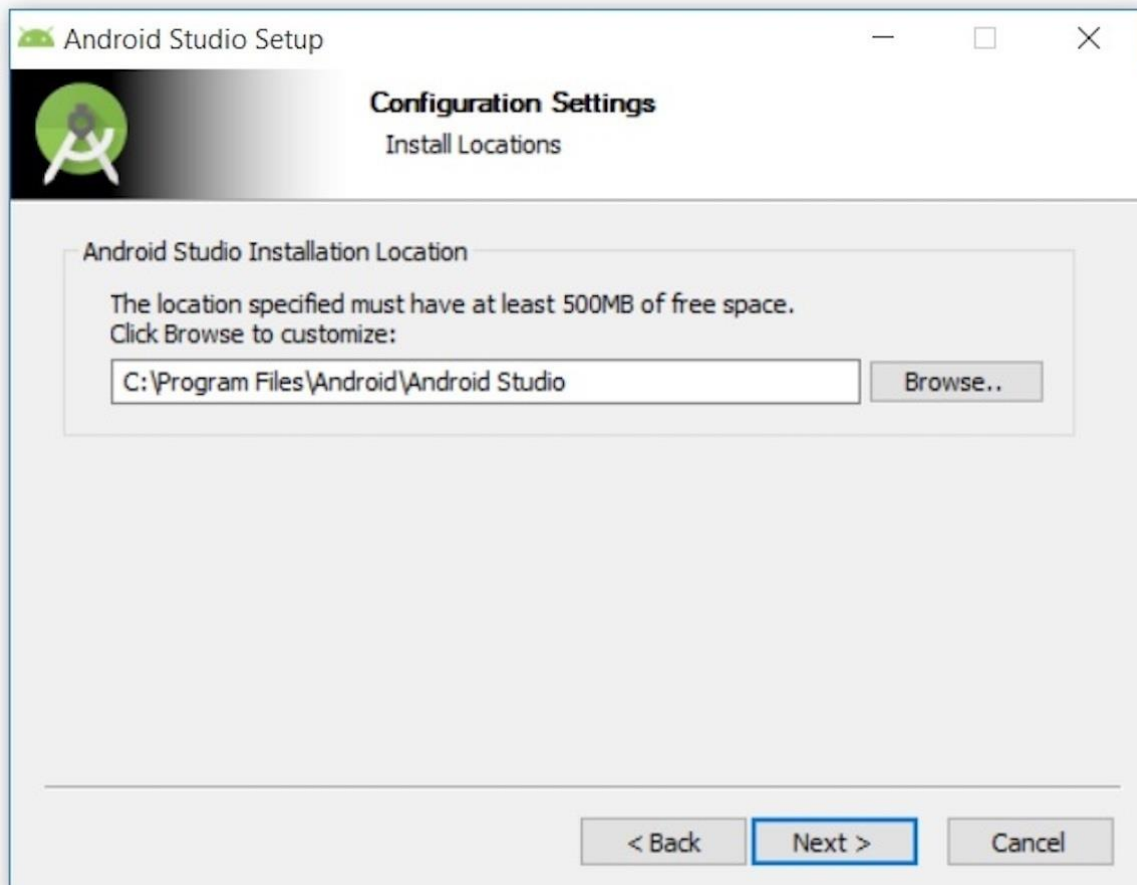


Figure 3. The installation location must have at least 500 MB free space

I kept the default installation location and clicked **Next**, and was greeted with the **Choose Start Menu Folder** panel.

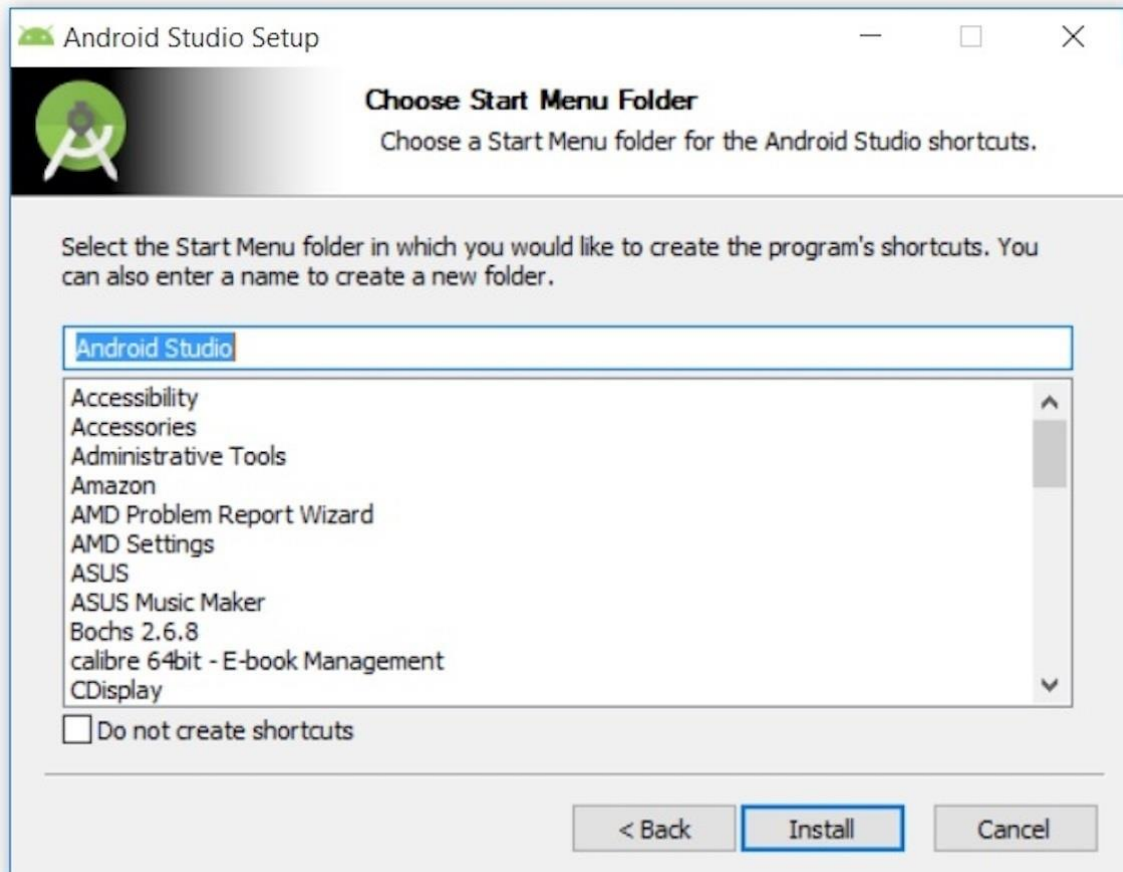


Figure 4. Select the folder in which to store Android Studio shortcuts

I kept the default setting and clicked **Install**. The following **Installing** panel appeared:

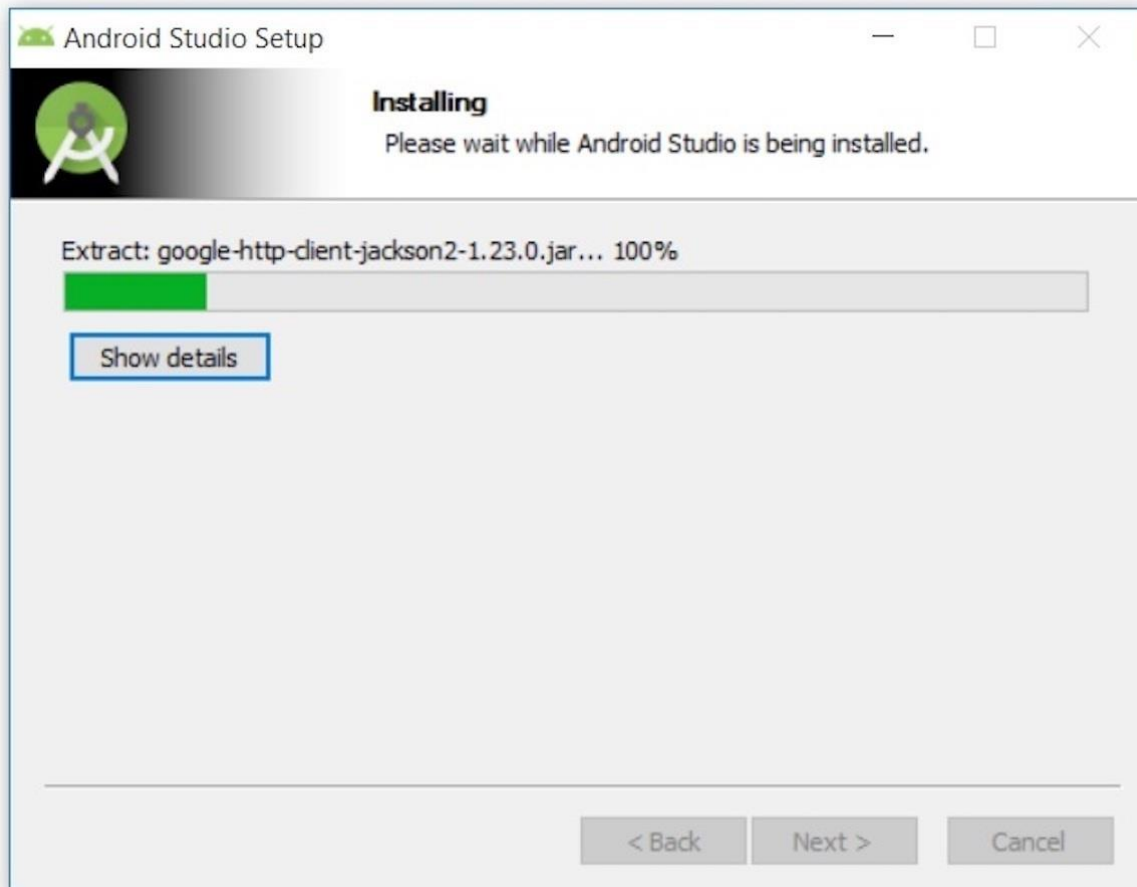


Figure 5. This panel shows the progress of the installation

Clicking **Show details** causes the names of files being installed and other activities to be displayed. When installation finished, the **Installation Complete** panel appeared.

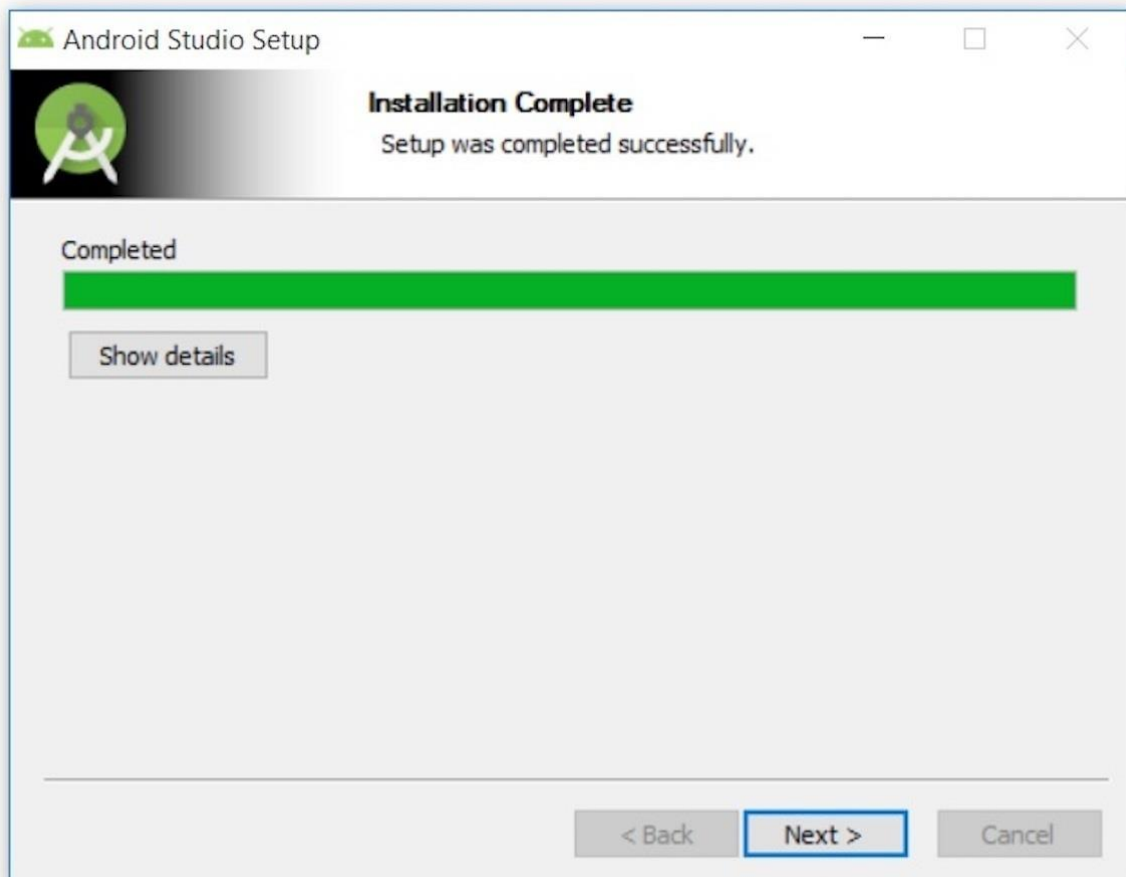


Figure 6. The Next button is enabled when installation completes

After clicking **Next**, the installer presented the **Completing Android Studio Setup** panel.

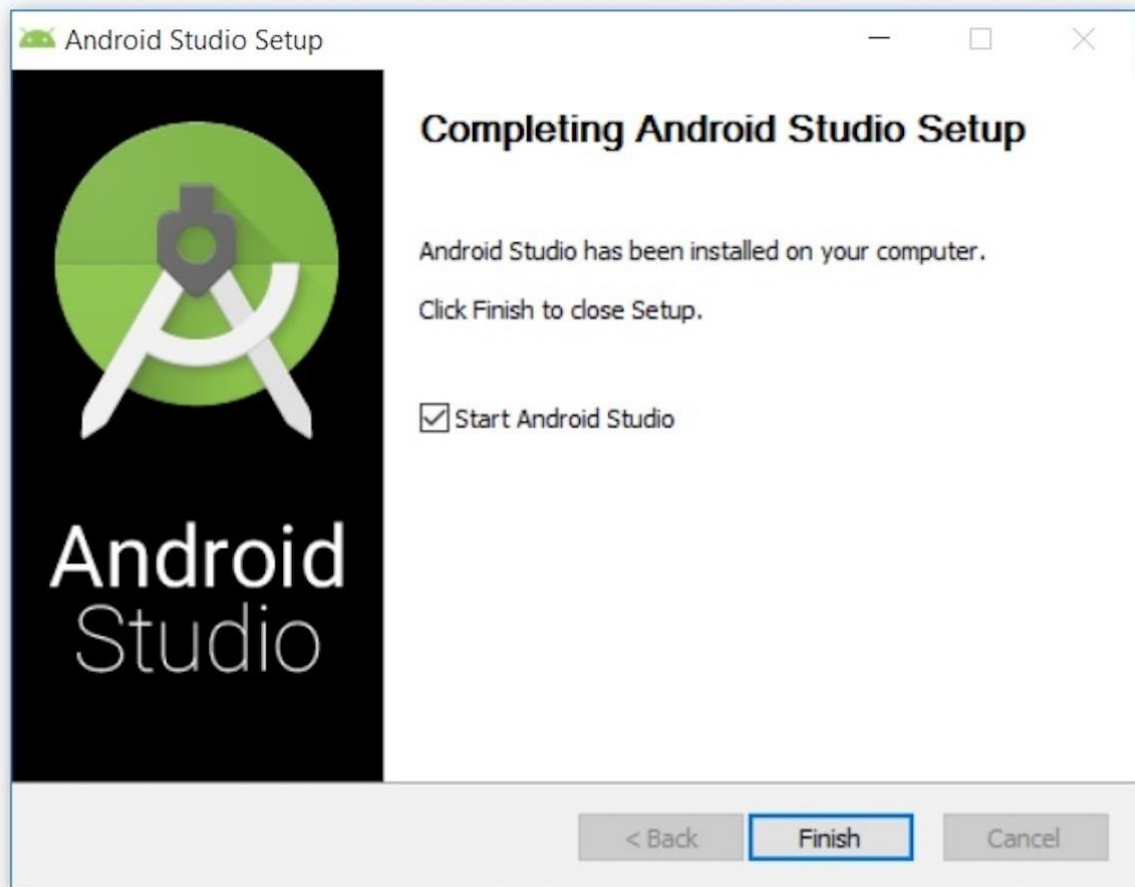


Figure 7. Leave the Start Android Studio checkbox checked to run this software

To complete the installation, I left the **Start Android Studio** box checked and clicked **Finish**.

Running Android Studio

The first time Android Studio runs, it presents a **Complete Installation** dialog box that offers the option of importing settings from a previous installation.

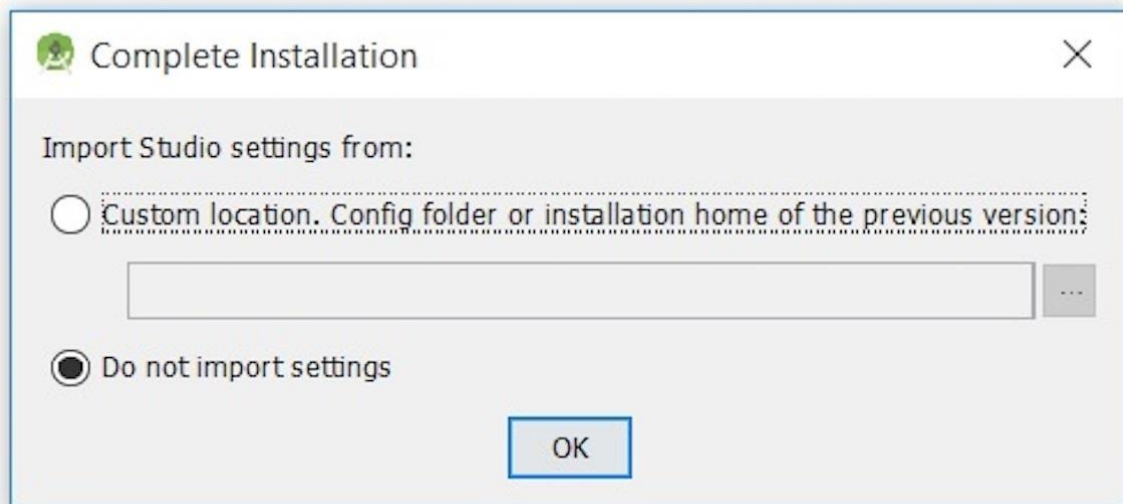


Figure 8. A previous installation's settings can be imported

I chose not to import settings (the default selection) and clicked **OK**, and was rewarded with the following splash screen:



Figure 9. Android Studio's splash screen

I also observed the following **Finding Available SDK Components** message box.

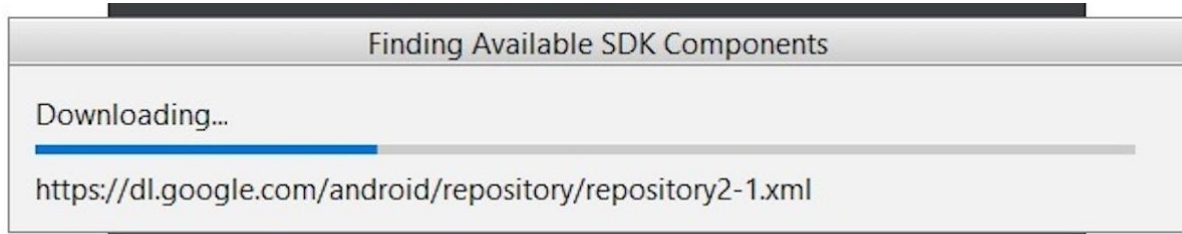


Figure 10. Android Studio downloads any SDK components that are needed (and available)

At this point, Android Studio presented the following **Android Studio Setup Wizard** dialog box:

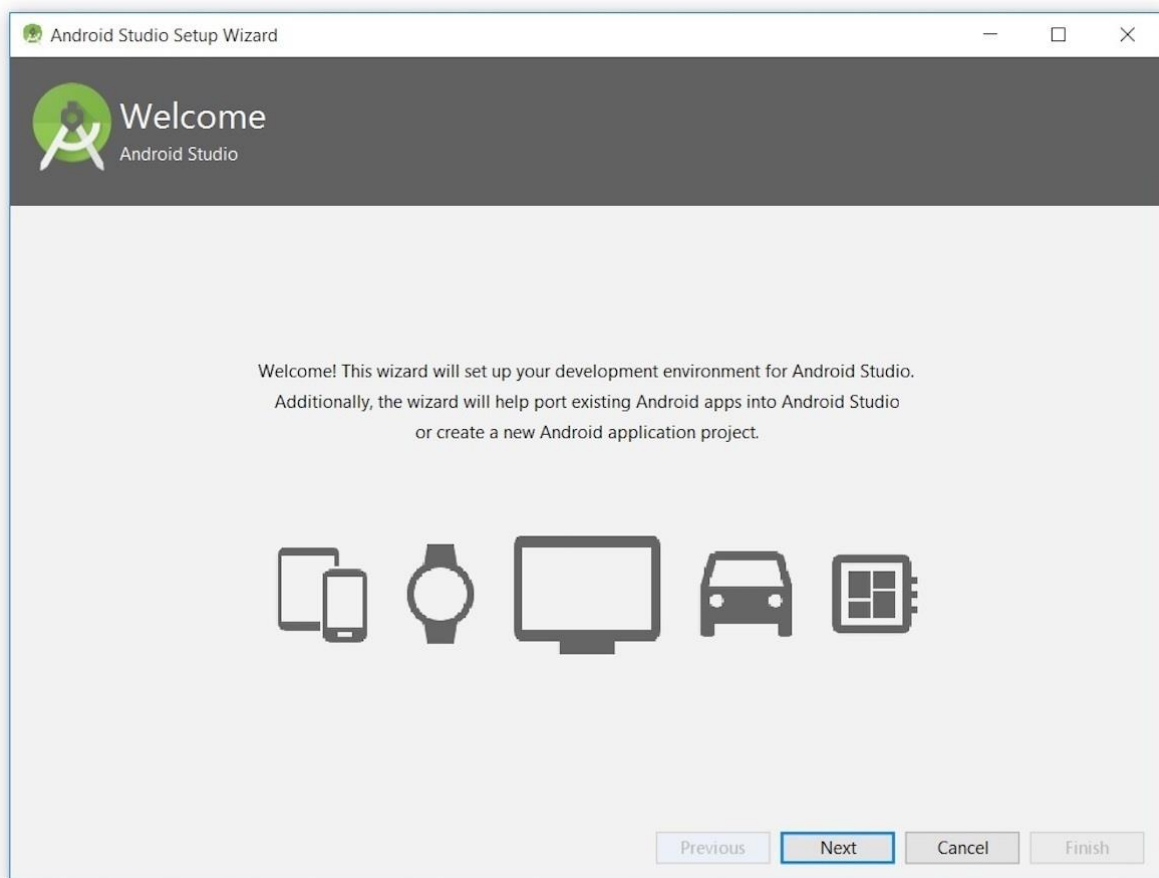


Figure 11. The wizard provides setup and app-porting capabilities

I clicked **Next**, and the wizard invited me to select an installation type. I kept the default standard setting.

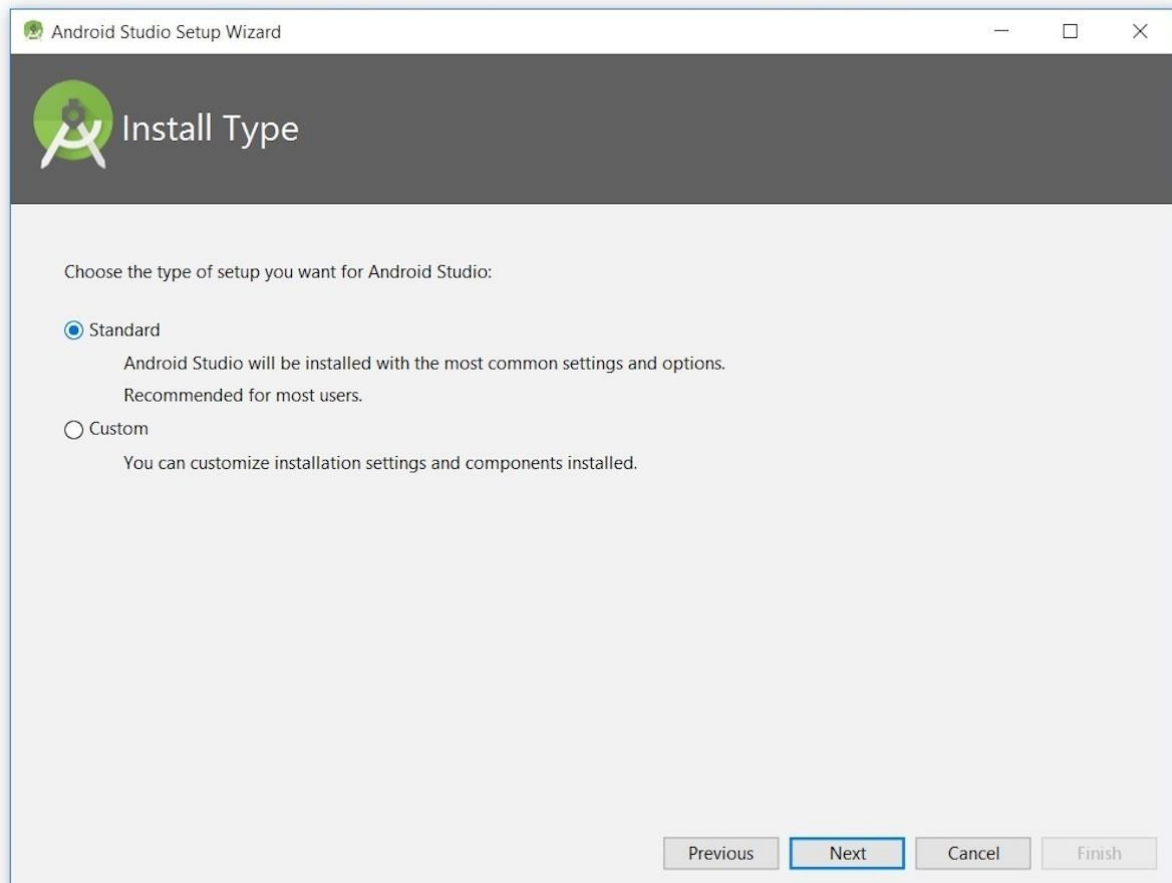


Figure 12. Choose an installation type

I was then given the opportunity to choose a user interface theme.

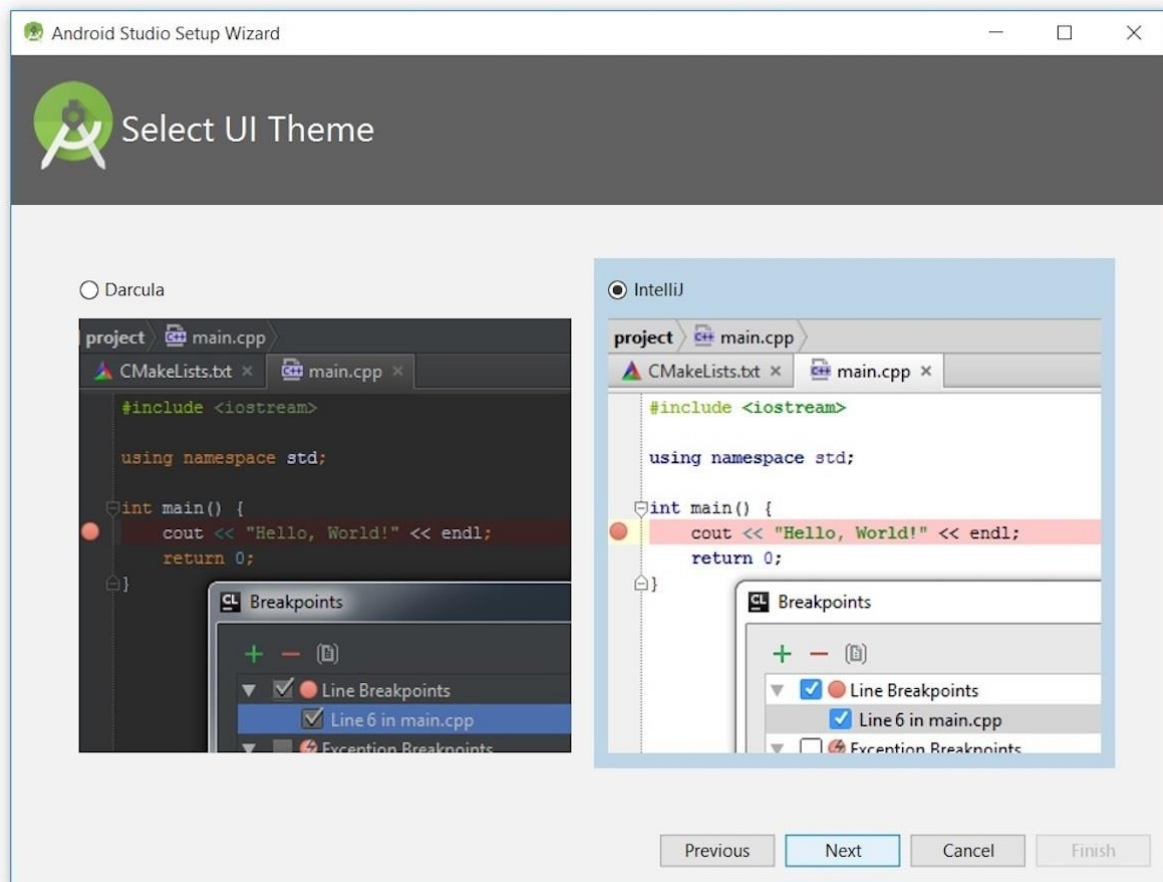


Figure 13. Put the bite on Android Studio by choosing the Darcula theme

I kept the default **IntelliJ** setting and clicked **Next**. Android Studio next provided the opportunity to verify settings.

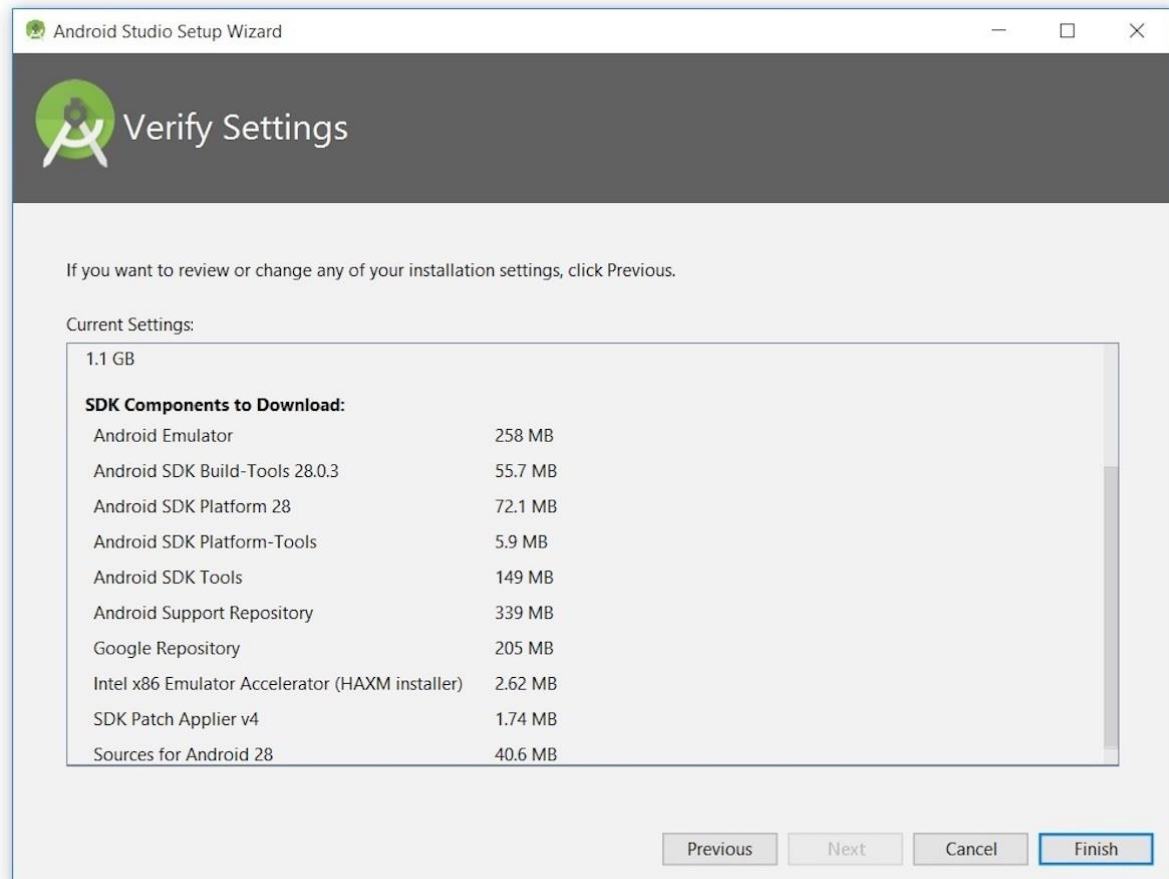


Figure 14. Android Studio identifies additional SDK components that will be downloaded
(click to enlarge)

I clicked **Finish** and Android Studio began the process of downloading SDK components.

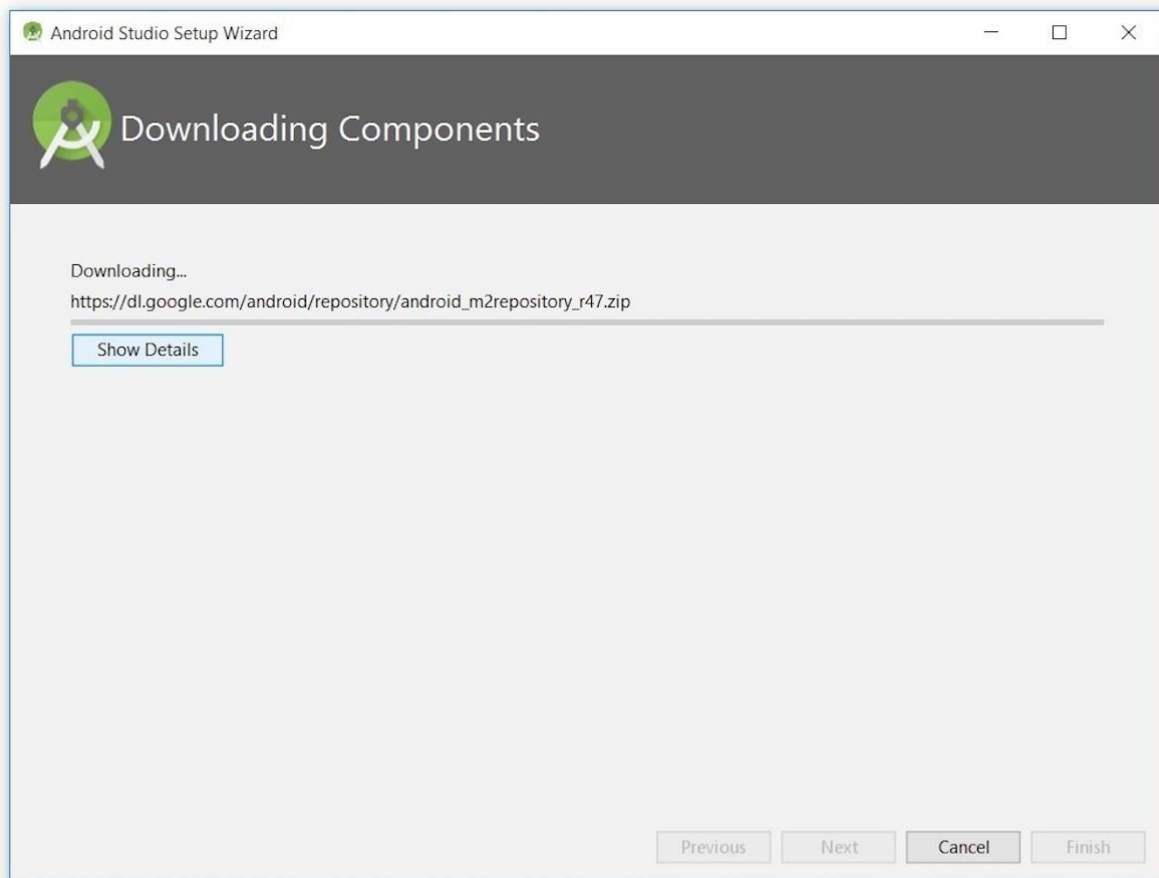


Figure 15. The wizard downloads and unzips SDK components

It can take several minutes for this part of the setup to finish. Clicking **Show Details** might relieve some boredom by revealing the various files being downloaded and unzipped.

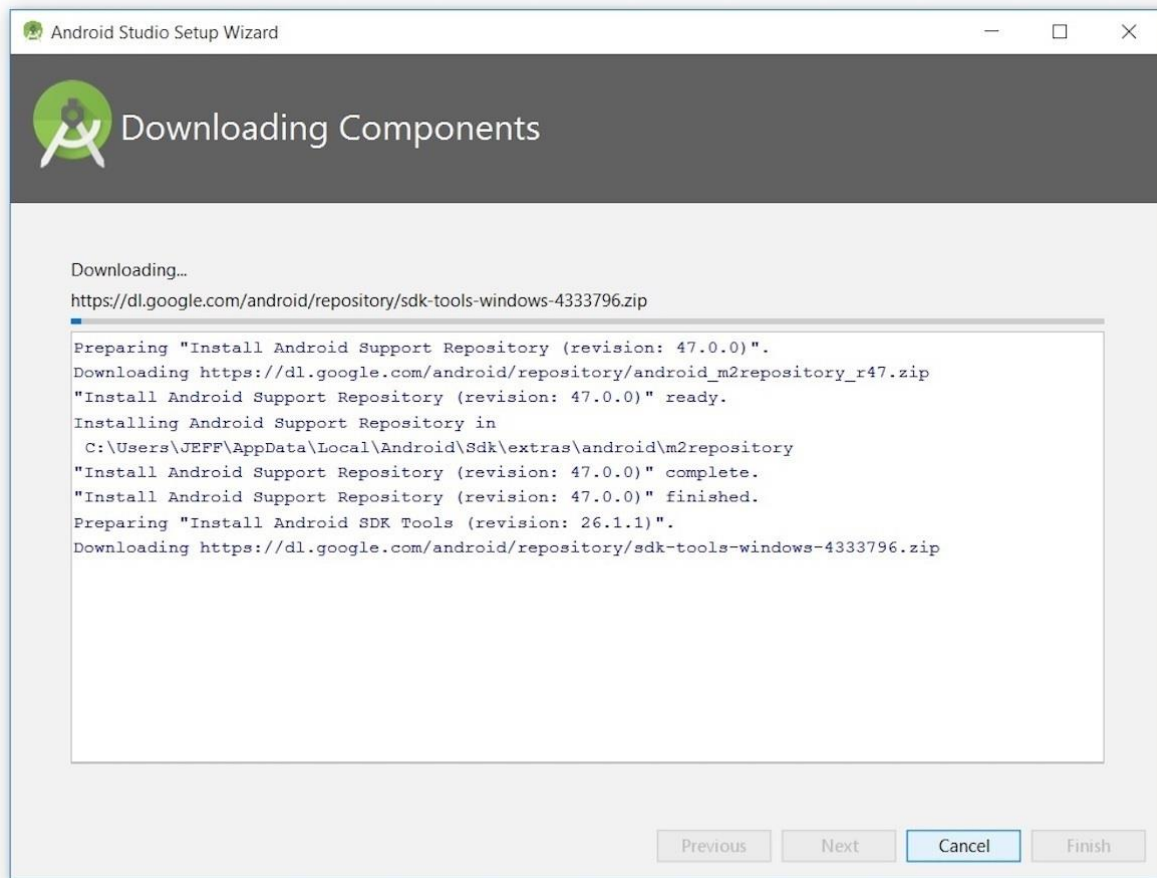


Figure 16. The wizard identifies the various archives being downloaded

For my AMD-based computer, an unpleasant surprise awaited after the components had completely downloaded and unzipped:

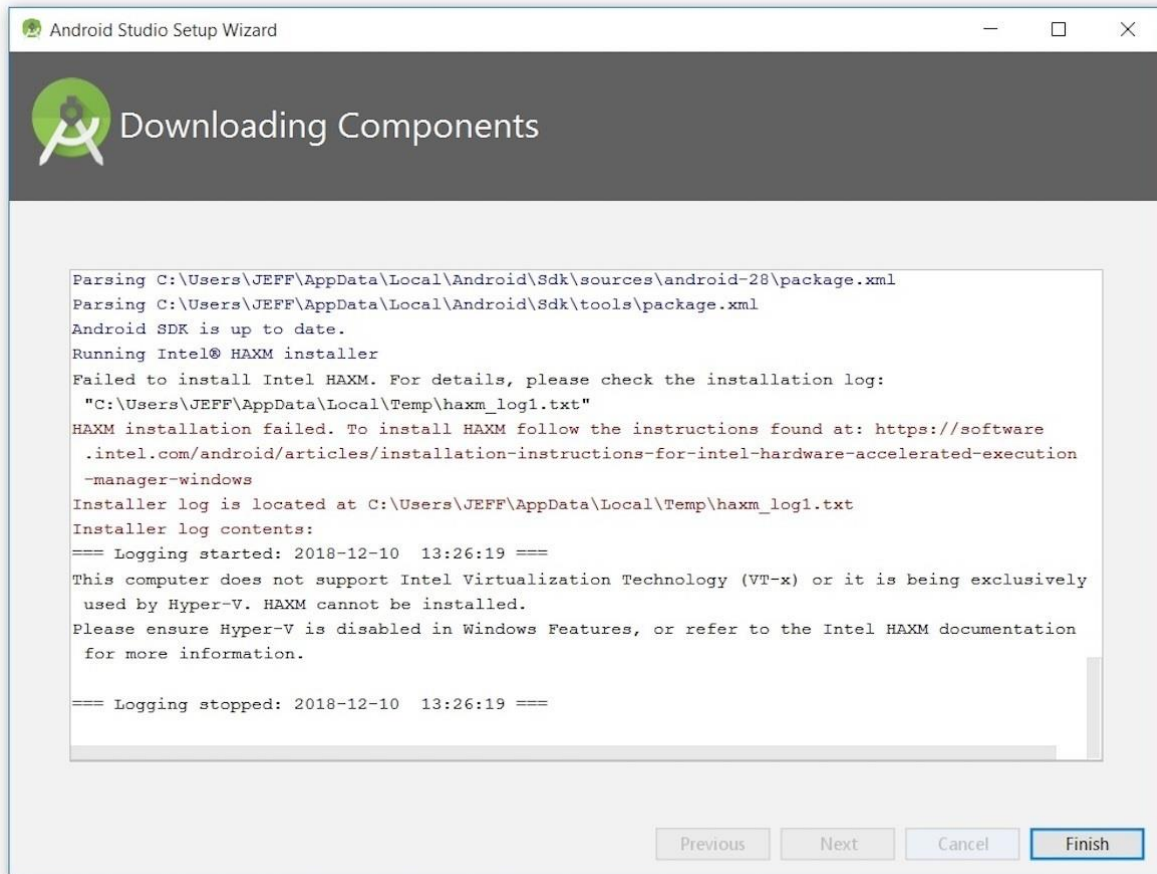


Figure 17. Intel-based hardware acceleration is unavailable

My options are to either put up with the slow emulator or use an Android device to speed up development. In Part 3 I'll show you how I resolved this issue.

Finally, I clicked **Finish** to complete the wizard. The **Welcome to Android Studio** dialog box appeared.

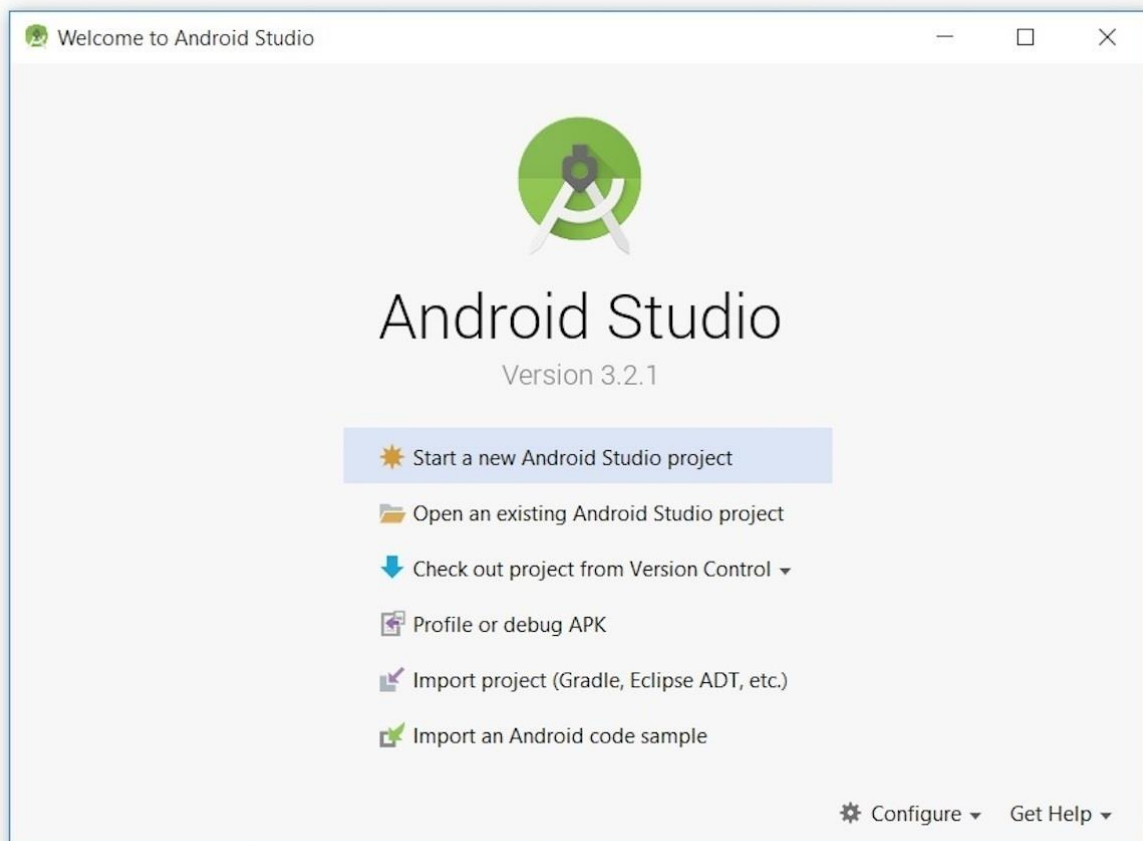


Figure 18. Welcome to Android Studio

This dialog box is used to start up a new Android Studio project, work with an existing project, and more. It can be accessed by selecting **Android Studio** from the Windows **Start** menu, or the equivalent on another platform.

Your first Android Studio mobile app

The quickest way to get to know Android Studio is to use it to develop an app. We'll start with a variation on the "Hello, World" application: a little mobile app that displays a "Welcome to Android" message.

In the steps that follow, you'll start a new Android Studio project and get to know the main window, including the editor window that you'll use to code the app in Part 2.

Starting a new project

From our setup so far, you should still have Android Studio running with the **Welcome to Android Studio** dialog box. From here, click **Start a new Android Studio project**. Android Studio will respond with the **Create New Project** dialog box shown in Figure 19.

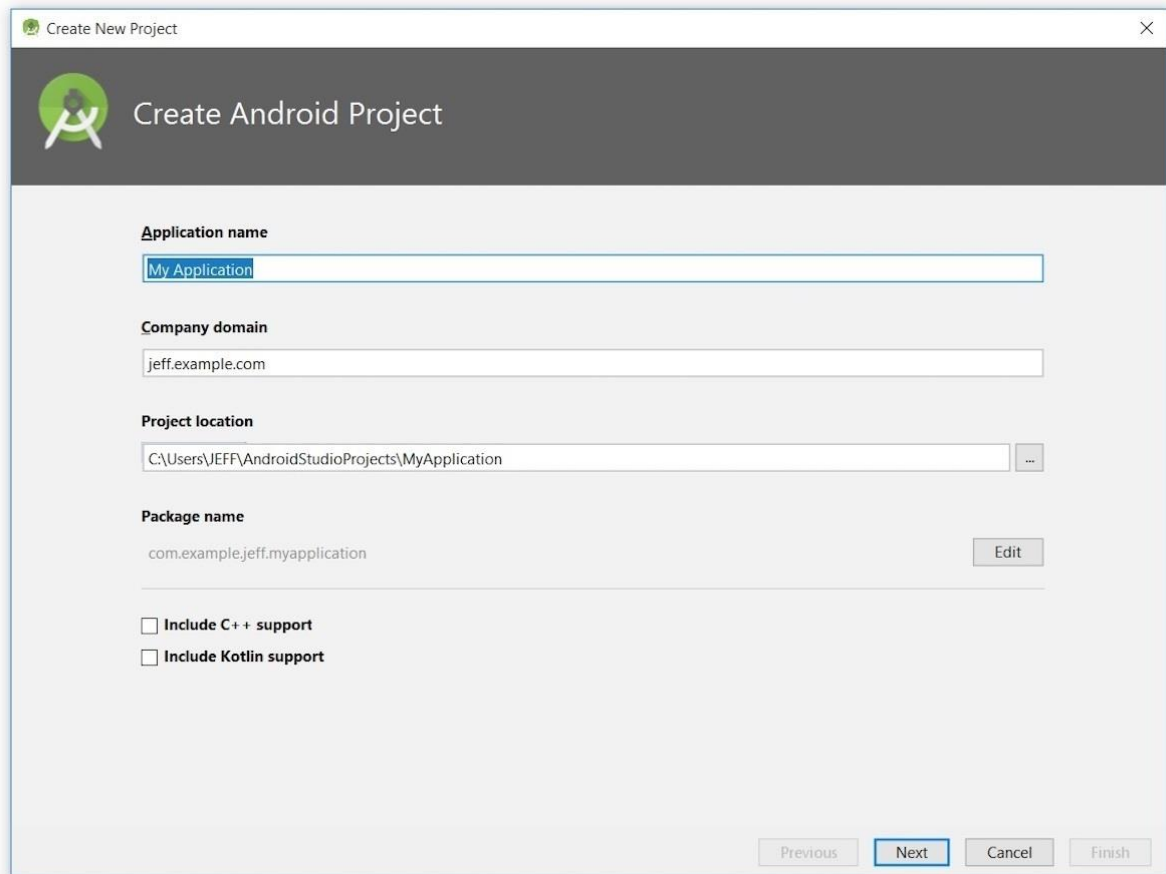


Figure 19. Create a new Android project

Enter **W2A** (Welcome to Android) as the application name and **javajeff.ca** as the company domain name. On my desktop, I observed **C:\Users\JEFF\AndroidStudioProjects\W2A** as the project location. Click **Next** to select your target devices.

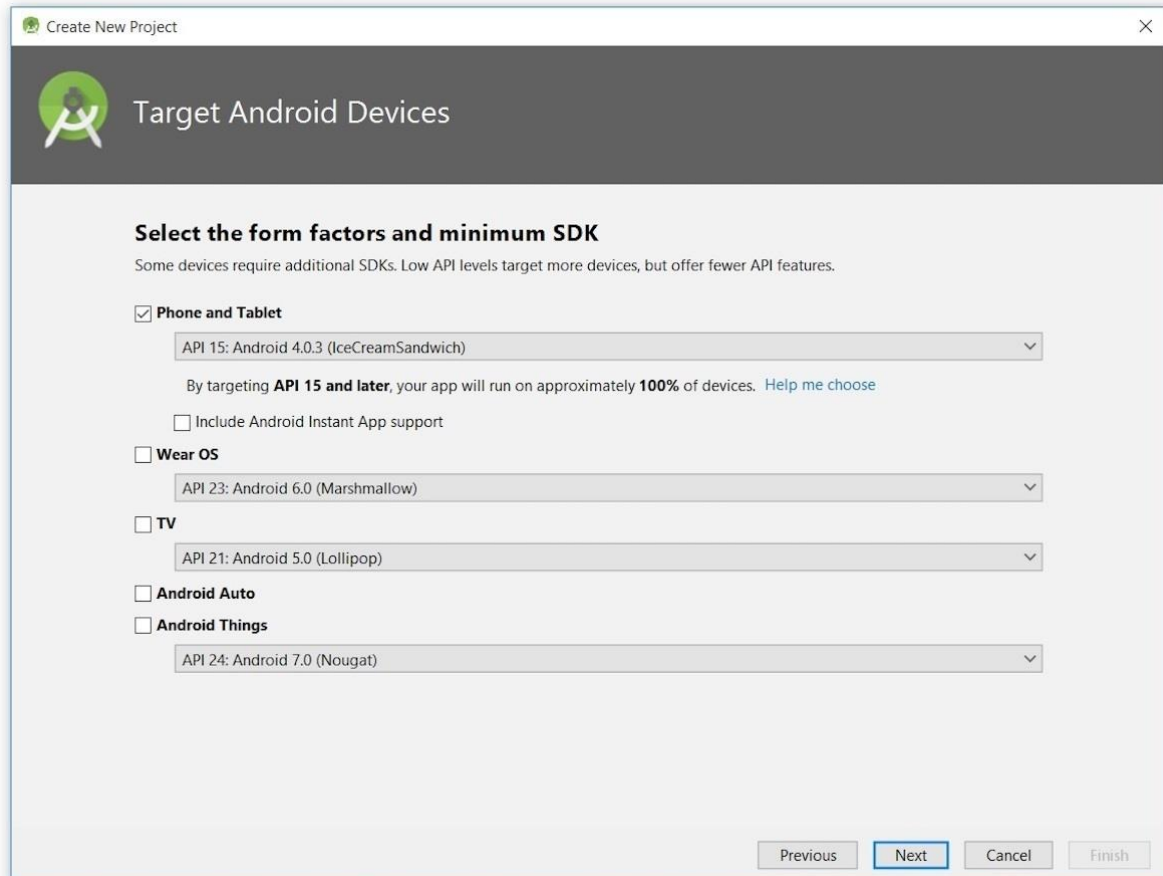


Figure 20. Select your target device categories

Android Studio lets you select *form factors*, or categories of target devices, for every app you create. I kept the default setting.

Click **Next**, and you will be given the opportunity to choose a template for your app's main activity. For now we'll stick with **Empty Activity**. Select this template (if necessary) and click **Next**.

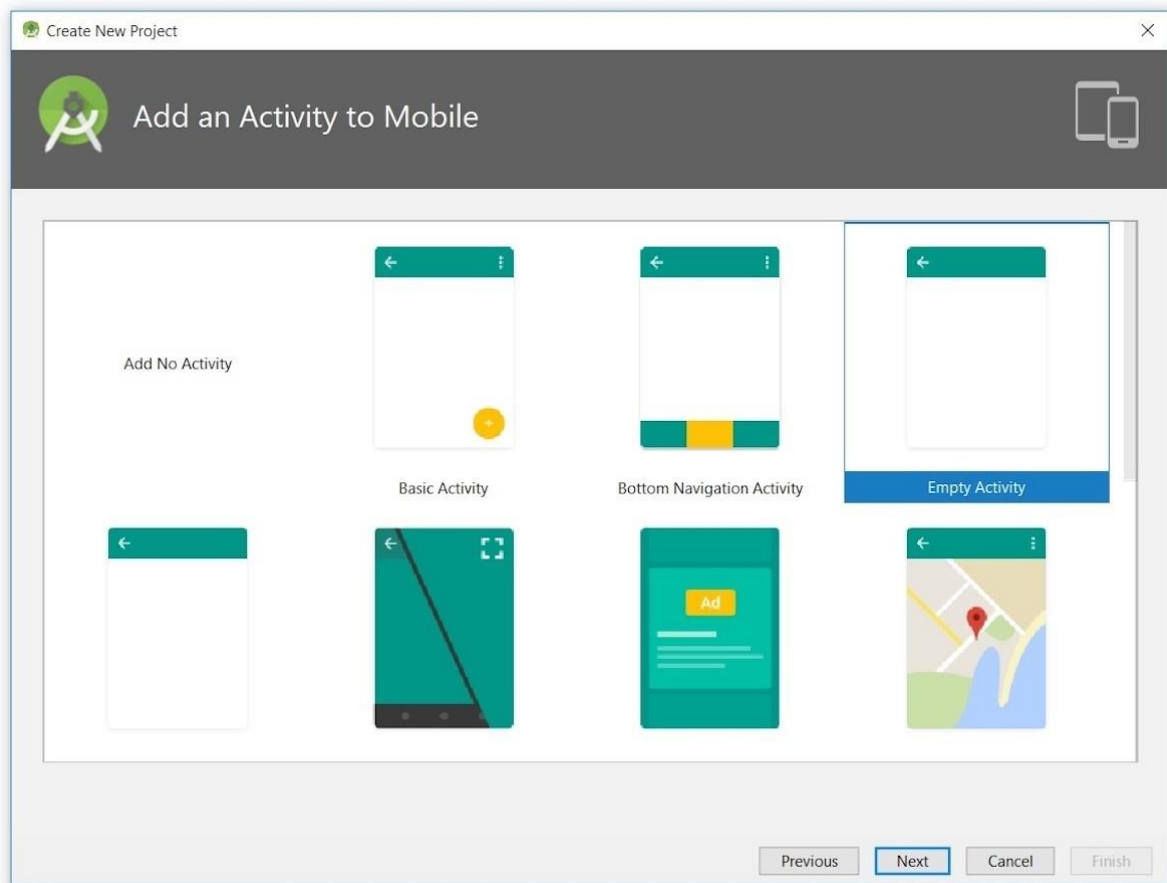
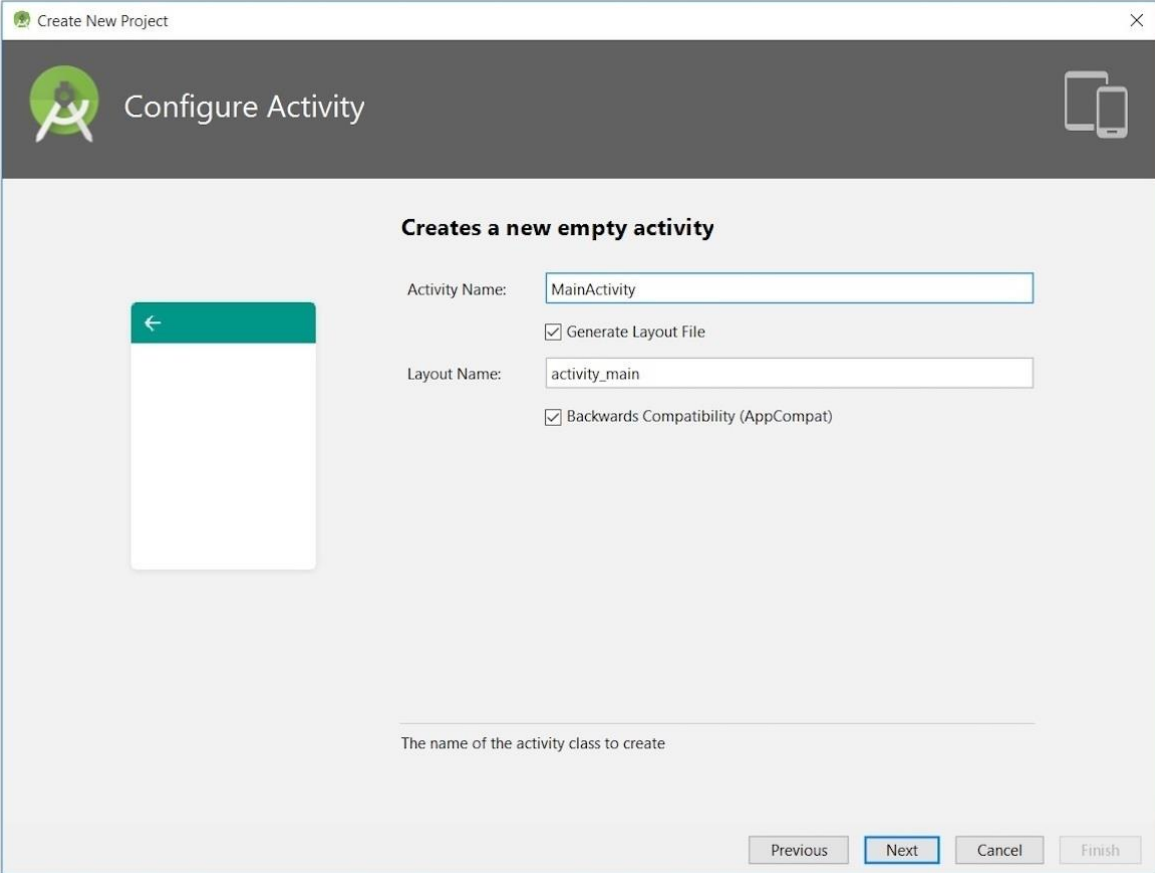


Figure 21. Specify an activity template

Next you'll customize the activity:



The screenshot shows the 'Configure Activity' dialog box in an IDE. The title bar says 'Create New Project'. The dialog has a dark header with the Android Studio logo and the text 'Configure Activity'. On the right of the header is a close button (X) and an icon of a tablet and phone. The main area is light gray and contains the heading 'Creates a new empty activity'. On the left is a preview of a simple activity layout with a teal header bar containing a white back arrow. To the right of the preview are two text input fields: 'Activity Name:' with 'MainActivity' and 'Layout Name:' with 'activity_main'. Between these fields are two checked checkboxes: 'Generate Layout File' and 'Backwards Compatibility (AppCompat)'. Below the input fields is a horizontal line with the text 'The name of the activity class to create' underneath it. At the bottom right are four buttons: 'Previous', 'Next' (highlighted with a blue border), 'Cancel', and 'Finish'.

Figure 22. Customize your activity

Enter **W2A** as the activity name and **main** as the layout name, and click **Next** to complete this step.

Reconfigured buttons

The next time you create an app for the chosen target device category, you'll probably discover that **Next** is disabled and **Finish** is enabled.

The first time you use Android Studio, you'll discover that it has to download some files related to its [constraint layout](#), which is used to build responsive user interfaces:

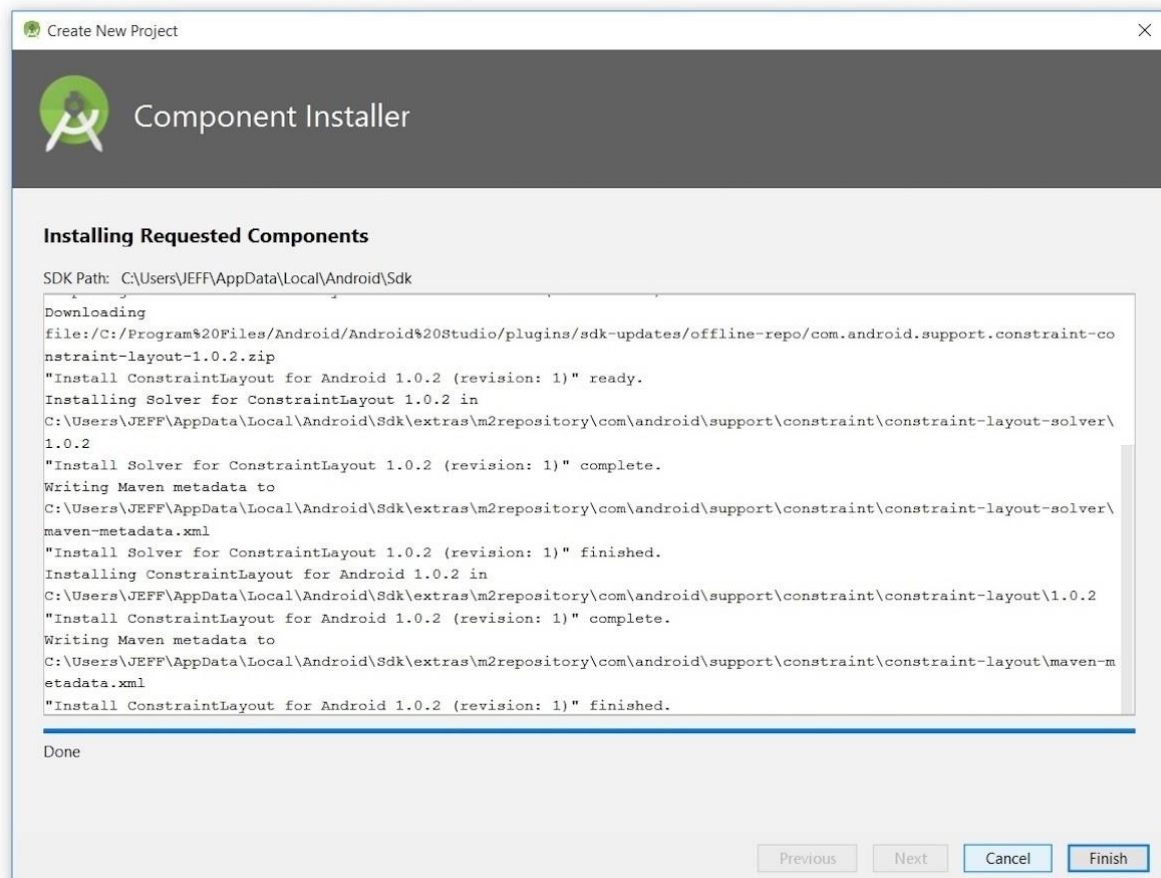


Figure 23. Constraint layout is the default layout used by Android Studio for new app projects

Android Studio enables **Finish** after downloading the constraint layout files. Click this button and Android Studio takes you to the main window.

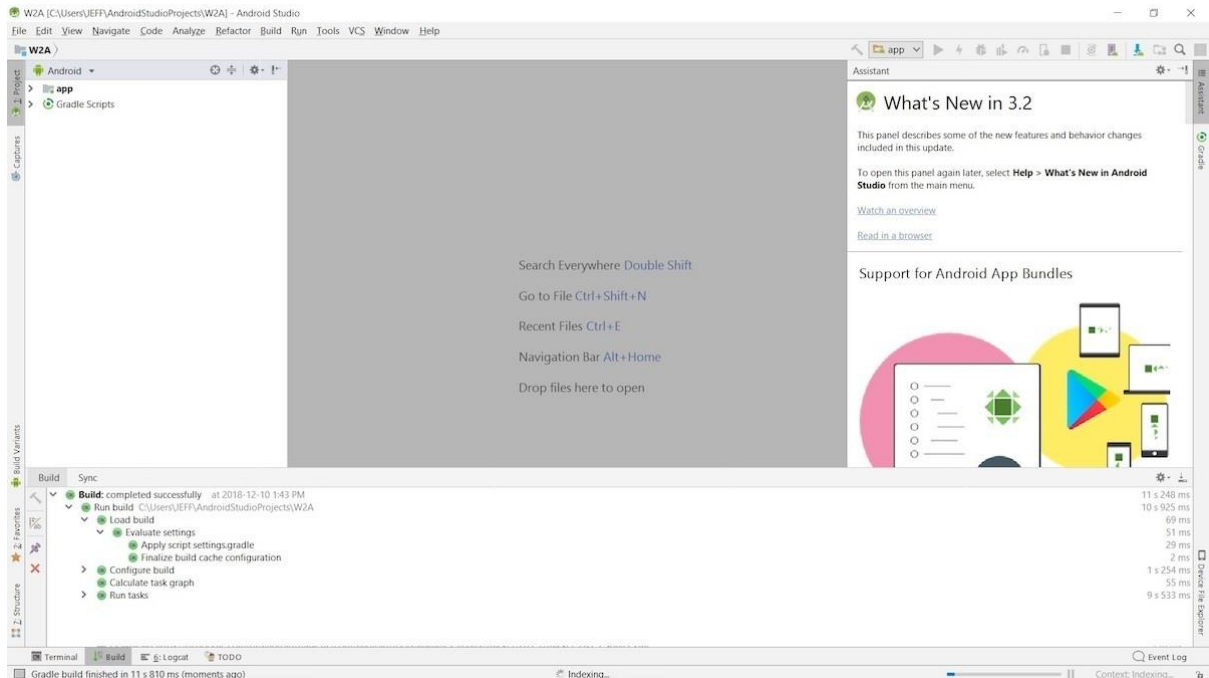


Figure 24. Android Studio's main window reveals that it has built a skeletal W2A app

The main window is divided into a menu bar and several other areas, which are identified in Figures 25 and 26. (Note that Figures 25 and 26 are courtesy of [Google](#).)

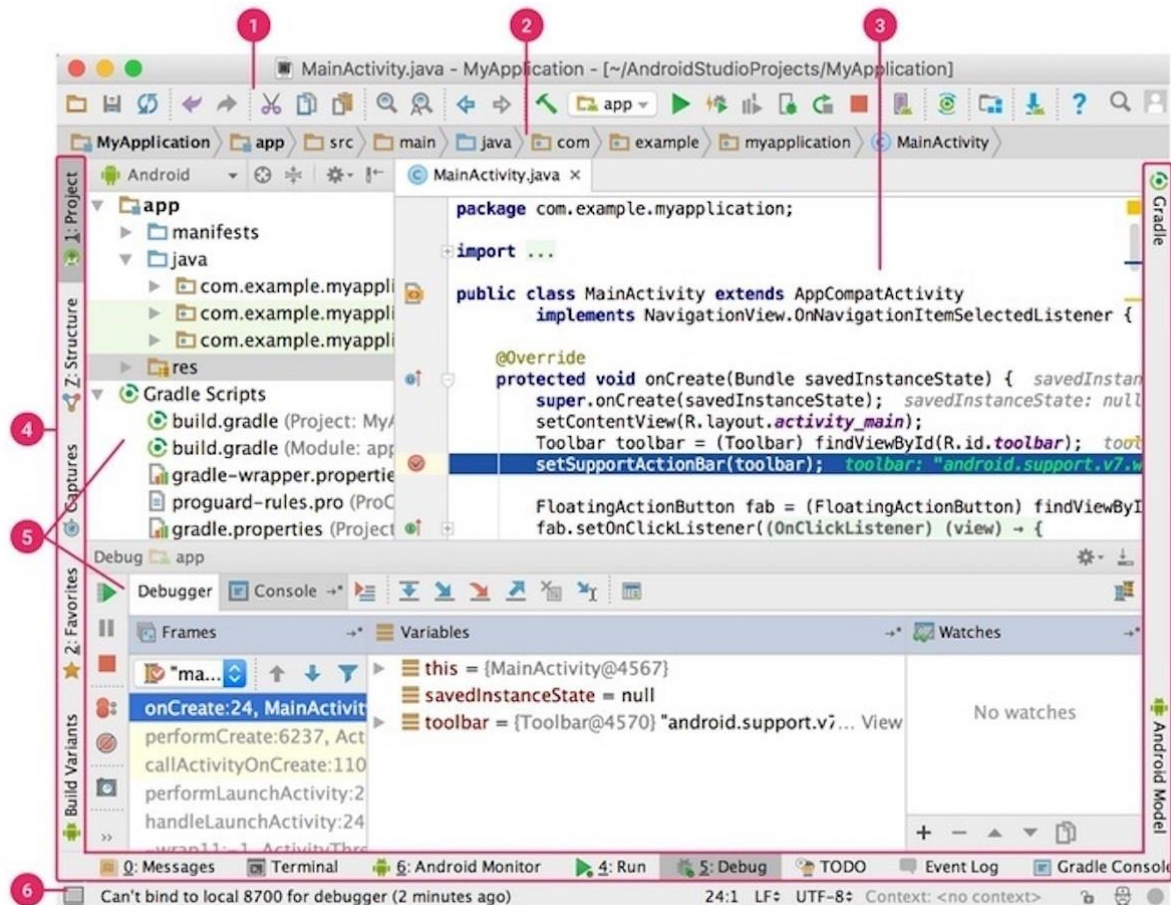


Figure 25. The different areas that comprise the main window

Check out the [Meet Android Studio](#) page to learn more about Android Studio's user interface.

Accessing AVD Manager and SDK Manager

To access the traditional AVD Manager or SDK Manager, select **AVD Manager** or **SDK Manager** from Android Studio's **Tools** menu.

The Project and editor windows

When you enter the main window (see Figure 24), you observe the Project window showing only **app** and **Gradle Scripts**. You'll have to expand the **app** branch of the project tree to observe more details.

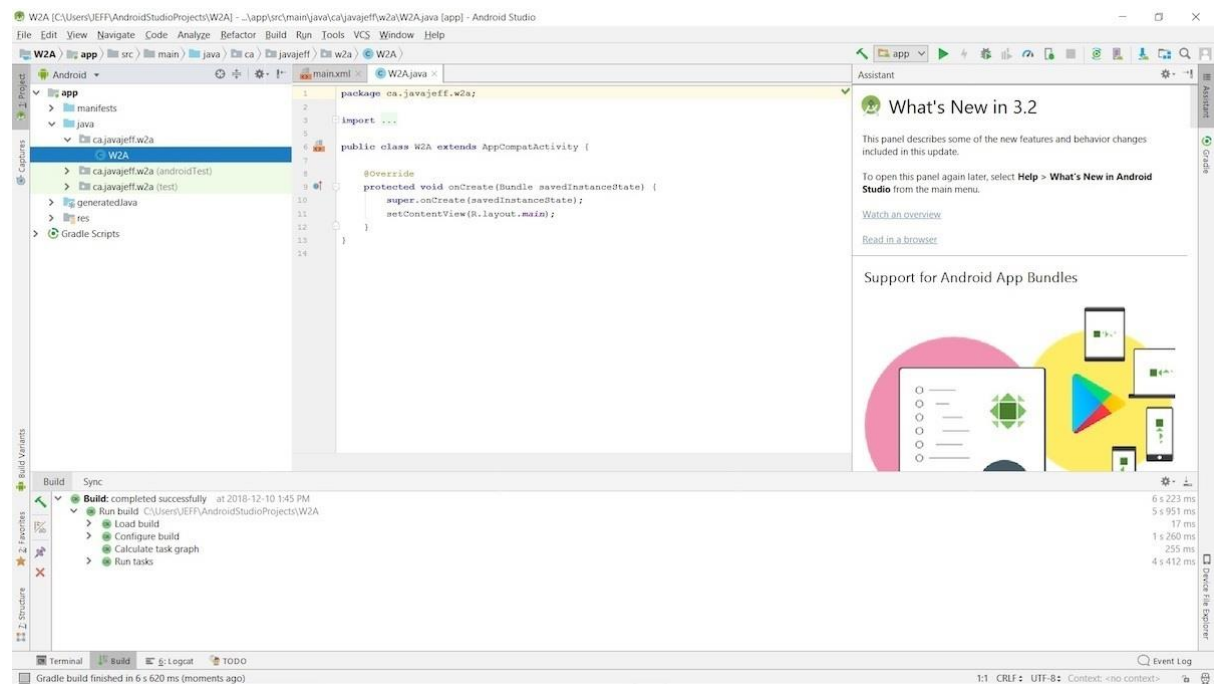


Figure 27. The Project window and an editor window show the skeletal W2A activity source code

The Project window is organized into a tree whose main branches are **app** and **Gradle Scripts**. The **app** branch is further organized into **manifests**, **java**, **generatedJava**, and **res** subbranches:

- **manifests** stores `AndroidManifest.xml`, which is an XML file that describes the structure of an Android app. This file also records permission settings (where applicable) and other details about the app.
- **java** stores an app's Java source files according to a package hierarchy, which is `ca.java.jeff.w2a` in this example. It also organizes files for testing purposes.
- **res** stores an app's resource files, which are organized into **drawable**, **layout**, **mipmap**, and **values** subbranches:
 - **drawable** is a mostly empty location in which to store an app's artwork; initially, the XML files for the launcher foreground and background **adaptive icons** are stored here.
 - **layout** is a location containing an app's layout files; `main.xml` (the main activity's layout file) is initially stored here.
 - **mipmap** is a location containing various `ic_launcher.png` files, which store launcher screen icons of different resolutions.
 - **values** is a location containing `colors.xml`, `strings.xml`, and `styles.xml`.
 - The **Gradle Scripts** branch identifies various `.gradle` (such as `build.gradle`) and `.properties` (such as `local.properties`) files that are used by Android Studio's Gradle-based build system.

- **Branch names and directory/file names**

- Each branch/subbranch corresponds to a directory name or to a file name. For example, **res** corresponds to the `res` directory and **strings.xml** corresponds to the `strings.xml` file.

CHAPTER 5

CODING

5.1 INTERMEDIATE CODE

[privlok/Main.activity.java](#)

(In this activity we have use splashed screen with the Application name for 5 sec and then this activity automatically navigate to the second activity that is home page after 5 sec.)

```
public      class
MainActivity
extends
AppCompatActivity
{
    @Override
    Protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```

        ImageView imageview =findViewById(R.id.imageView);

        Animation animation =
        AnimationUtils.loadAnimation(getApplicationContext(),R.anim.fade);

        imageview.startAnimation(animation);

        Thread timer = new Thread(){

            @Override
            public void run() {

                try{

                    sleep(8000);

                    Intent i = new
                    Intent(getApplicationContext(),BottomNavigationActivity.class);

                    startActivity(i);

                    finish();

                    super.run();

                }catch (InterruptedException e){

                    e.printStackTrace();

                }

            }

        };

        timer.start();

    }

}

```

[privlok](#)/add activity

(In this activity we add all details or pasword that the user wants to add.)


```

public class
AddPasswords extends
AppCompatActivity {

    private TextInputLayout textInputTitle;
    private TextInputLayout textInputPassword;
    private TextInputLayout textInputAccount;
    private TextInputLayout textInputUsername;
    Button save;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_passwords);

        textInputTitle = findViewById(R.id.text_input_title);
        textInputPassword = findViewById(R.id.text_input_password);
        textInputAccount = findViewById(R.id.text_input_account);
        textInputUsername = findViewById(R.id.text_input_username);
        save = findViewById(R.id.btnInsert);
    }

    private boolean validateTitle() {
        String Title =
textInputTitle.getText().toString().trim();

        if(Title.isEmpty()){
            textInputTitle.setError("Field can't be empty");
            return false;
        } else {
            textInputTitle.setError(null);
            return true;
        }
    }
}

```

[privlok](#)/add activity.xml

```
<com.google.android.material.textfield.TextInputLayout
```

```
    android:id="@+id/text_input_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    app:errorEnabled="true">
```

```
<com.google.android.material.textfield.TextInputEditText
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
        android:hint="Title"
        android:textSize="30dp"
        android:background="#00000000"
        android:inputType="text"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
```

```
    android:id="@+id/text_input_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:counterEnabled="true"
        app:counterMaxLength="20"
        android:layout_margin="20dp"
        app:errorEnabled="true">
```

```
<com.google.android.material.textfield.TextInputEditText
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
        android:hint="Username"
        android:textSize="20dp"
        android:background="#00000000"
```

```
android:inputType="text"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
```

```
    android:id="@+id/text_input_account"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="20dp"  
    app:counterEnabled="true"  
    app:counterMaxLength="16"  
    app:errorEnabled="true">
```

```
<com.google.android.material.textfield.TextInputEditText
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"  
    android:hint="Account Number"  
    android:textSize="20dp"  
    android:background="#00000000"  
    android:inputType="number"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
```

```
    android:id="@+id/text_input_password"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="20dp"  
    app:errorEnabled="true"  
    app:passwordToggleEnabled="true">
```

```
<com.google.android.material.textfield.TextInputEditText
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"  
    android:hint="Password"  
    android:background="#00000000"  
    android:textSize="20dp"
```

```
    android:inputType="textPassword"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<Button  
    android:id="@+id/btnInsert"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="20dp"  
    android:text="Save"  
    android:textSize="20dp"  
    android:onClick="saveInput" />
```

[privlok](#)/bottom activity.xml

```
<androidx.constraintlayout.widget.ConstraintLayout  
xmlns:android=http://schemas.android.com/apk/res/android
```

```
xmlns:app="http://schemas.android.  
com/apk/res-auto"
```

```
xmlns:tools="http://schemas.android.com/tools"  
android:layout_width="match_parent"  
android:layout_height="match_parent"  
tools:context=".BottomNavigationActivity">
```

```
<ListView  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent"
```

```
app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"
```

```

        app:layout_constraintTop_toTopOf="parent" />

<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:id="@+id/containerLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.8"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0">

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/floating_action_button"
        android:layout_width="68dp"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|right"
        android:layout_margin="32dp"

        android:src="@drawable/ic_baseline_add_24"
        app:backgroundTint="@color/teal_200"
        app:fabSize="normal"
        app:rippleColor="@color/teal_700" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

[privlok](#)/bottom navigation activity

```

public
class
BottomN
avigation
onActiv
ity
extends
AppComp
atActiv
ity {

    @Override

```

```

        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_bottom_navigation);
            FloatingActionButton floatingActionButton=
findViewById(R.id.floating_action_button);
            floatingActionButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent i = new
Intent(getApplicationContext(),AddPasswords.class);
                    startActivity(i);
                }
            });
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.popmenu,menu);
            return super.onCreateOptionsMenu(menu);
        }
    }
}

```

[privlok/mainactivity.xml](#)

```

<RelativeLayout
xmlns:android="http://schemas.andr
oid.com/apk/res/android"

```

```

xmlns:app="http://schemas.android.com/ap
k/res-auto"

```

```

xmlns:tools="http://schemas.android.com/
tools"

```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity"

```

```

        android:background="@drawable/gradient_b
ackground"

```

```

        android:theme="@style/Theme.MaterialComp
onents.DayNight.NoActionBar">

```

```

            <ImageView
                android:id="@+id/imageView"

```

```

            android:layout_width="match_parent"

```

```

            android:layout_height="wrap_content"
            android:layout_marginTop="180dp"

```

```

        android:layout_centerInParent="true"

        app:layout_constraintEnd_toEndOf="parent"
        "

        app:layout_constraintHorizontal_bias="0.
        0"

        app:layout_constraintStart_toStartOf="pa
        rent"

        app:layout_constraintTop_toTopOf="parent
        "

        app:srcCompat="@drawable/lock"
    />

</RelativeLayout>

```

5.2 UPDATE CODE

[privlok](#)/(UPDATE PROJECT CODE) AddPassword.java

```

publi
c
class
AddPa
sswor
ds
exten
ds
AppCo
mpatA
ctivi
ty {

    private TextInputLayout textInputTitle;
    private TextInputLayout textInputPassword;
    private TextInputLayout textInputAccount;
    private TextInputLayout textInputUsername;

```

```

//EditText
textInputTitle,textInputPassword,textInputAccount,textInputUsername;
Button save;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_passwords);
    MyDataHelper mydb=new MyDataHelper(this);
    textInputTitle = findViewById(R.id.text_input_title);
    textInputPassword = findViewById(R.id.text_input_password);
    textInputAccount = findViewById(R.id.text_input_account);
    textInputUsername = findViewById(R.id.text_input_username);
    save = findViewById(R.id.btnInsert);

    save.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

//Toast.makeText(getApplicationContext(),textInputTitle.getText().
.getText().toString(),Toast.LENGTH_LONG).show();

            if(!validateTitle() && !validatePassword() &&
!validateAccount() && !validateUsername()) {

                long check=
mydb.insertRecord(textInputTitle.getText().getText().toString(),t
extInputUsername.getText().getText().toString(),textInputAccount.
getText().getText().toString(),textInputPassword.getText().ge
tText().toString());
                if(check == -1)
                {
                    //displayMessage("Error", "Record not
inserted");

Toast.makeText(getApplicationContext(),"Record not
inserted",Toast.LENGTH_LONG).show();
                }
                else
                {
                    //displayMessage("Info","Record is inserted
Successfully");

Toast.makeText(getApplicationContext(),"Record is inserted
Successfully",Toast.LENGTH_LONG).show();
                }
            }
            else{
                Toast.makeText(getApplicationContext(),"Please record
insert first",Toast.LENGTH_LONG).show();
            }
        }
    });
}
/*public void displayMessage(String title, String msg) {
    AlertDialog.Builder builder = new
AlertDialog.Builder(AddPasswords.this);

```



```

        builder.setTitle(title);
        builder.setMessage(msg);
        builder.setCancelable(true);
        AlertDialog alertDialog = builder.create();
        alertDialog.show();
    }*/

    private boolean validateTitle() {
        String Title =
textInputTitle.getText().getText().toString().trim();

        if(Title.isEmpty()){
            textInputTitle.setError("Field can't be empty");
            return true;
        }
        else {
            textInputTitle.setError(null);
            return false;
        }
    }

    private boolean validatePassword() {
        String Password =
textInputPassword.getText().getText().toString().trim();

        if(Password.isEmpty()){
            textInputPassword.setError("Field can't be empty");
            return true;
        } else {
            textInputPassword.setError(null);
            return false;
        }
    }

    private boolean validateAccount() {
        String Account =
textInputAccount.getText().getText().toString().trim();

        if(Account.length() > 16){
            textInputAccount.setError("Invalid Account Number");
            return true;
        } else {
            textInputAccount.setError(null);
            return false;
        }
    }

    private boolean validateUsername() {
        String Username =
textInputUsername.getText().getText().toString().trim();

        if(Username.length() > 20){
            textInputUsername.setError("Username is too long");
            return true;
        } else {
            textInputUsername.setError(null);
            return false;
        }
    }

```

```

    }
}

```

[privlok/BottomNavigationActivity.java\(update\)](#)

(In this activity we use the menu and fab (floating action button) this fab button is used to add the passwords with name and in this home page we show all the details we have added previously in our app with the card style.)

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bottom_navigation);
    recyclerView= findViewById(R.id.recyclerview);
    fab= findViewById(R.id.floating_action_button);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i = new
Intent(getApplicationContext(),AddPasswords.class);
            startActivity(i);
        }
    });
    myDb= new MyDataHelper(BottomNavigationActivity.this);
    //array_id = new ArrayList<>();
    array_title = new ArrayList<>();
    array_username = new ArrayList<>();
    array_account = new ArrayList<>();
    array_password = new ArrayList<>();
    storeDataInArrays();
    customAdapter = new
CustomAdapter(BottomNavigationActivity.this, array_title,
array_username, array_account,array_password);
    recyclerView.setAdapter(customAdapter);
    recyclerView.setLayoutManager(new
LinearLayoutManager(BottomNavigationActivity.this));
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.popmenu,menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

```

```

        if(item.getItemId() == R.id.delete_all){
            confirmDialog();
        }
        return super.onOptionsItemSelected(item);
    }

    /*@Override
    protected void onActivityResult(int requestCode, int
resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == 1){
            recreate();
        }
    }*/

    void storeDataInArrays(){
        Cursor cursor = myDb.readAllData();
        if(cursor.getCount() == 0){
            Toast.makeText(this,"No Data found",
Toast.LENGTH_LONG).show();
        }
        else{
            while(cursor.moveToNext()){
                array_title.add(cursor.getString(0));
                array_username.add(cursor.getString(1));
                array_account.add(cursor.getString(2));
                array_password.add(cursor.getString(3));
            }
        }
    }
}

```

[privlok](#)/CustomAdaptar.java(update)

(This is a adapter class for manipulating all the data that is used in this app.)

```
CustomAdapter(  
Context  
context,ArrayL  
ist  
arrayList_titl  
e, ArrayList  
arrayList_user  
name,  
ArrayList  
arrayList_acco  
unt, ArrayList  
arrayList_pass  
word)
```

```
{  
    this.context= context;  
    //this.activity=activity;  
    //this.arrayList_id = arrayList_id;  
    this.arrayList_title=arrayList_title;  
    this.arrayList_username=arrayList_username;  
    this.arrayList_account=arrayList_account;  
    this.arrayList_password=arrayList_password;  
}  
  
@NonNull  
@Override  
public MyViewHolder onCreateViewHolder(@NonNull  
ViewGroup parent, int viewType) {  
    LayoutInflater inflater =  
LayoutInflater.from(context);  
    View  
view=inflater.inflate(R.layout.list_layout_data,parent,false  
);  
    return new MyViewHolder(view);  
}  
//@RequiresApi(api = Build.VERSION_CODES.M)  
@Override  
public void onBindViewHolder(@NonNull MyViewHolder  
holder, int position) {  
  
holder.title_txt.setText(String.valueOf(arrayList_title.get(  
position)));  
  
holder.username_txt.setText(String.valueOf(arrayList_usernam  
e.get(position)));  
  
holder.account_txt.setText(String.valueOf(arrayList_account.  
get(position)));  
  
holder.password_txt.setText(String.valueOf(arrayList_passwor  
d.get(position)));  
  
    //Recyclerview onClickListener  
    holder.mainLayout.setOnClickListener(new  
View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            Intent intent = new Intent(context,
```

```

UpdateActivity.class);
        //intent.putExtra("id",
String.valueOf(arrayList_id.get(position)));
        intent.putExtra("title",
String.valueOf(arrayList_title.get(position)));
        intent.putExtra("username",
String.valueOf(arrayList_username.get(position)));
        intent.putExtra("account",
String.valueOf(arrayList_account.get(position)));
        intent.putExtra("password",
String.valueOf(arrayList_password.get(position)));
        context.startActivity(intent);
        //activity.startActivityForResult(intent,
1);
    }
    });
}

@Override
public int getItemCount() {
    return arrayList_title.size();
}

public class MyViewHolder extends
RecyclerView.ViewHolder {

    LinearLayout mainLayout;
    TextView title_txt, username_txt, account_txt,
password_txt;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        title_txt =
itemView.findViewById(R.id.textViewTitle);
        username_txt =
itemView.findViewById(R.id.textViewUsername);
        account_txt =
itemView.findViewById(R.id.textViewAccountNo);
        password_txt =
itemView.findViewById(R.id.textViewPassword);
        mainLayout =
itemView.findViewById(R.id.mainLayout);
        //Animate Recyclerview
        Animation translate_anim =
AnimationUtils.loadAnimation(context,
R.anim.translate_anim);
        mainLayout.setAnimation(translate_anim);
    }
}

```

[privlok](#)/bottomNavigationActivity.xml(update)

(In this activity we use the menu and fab (floating action button) this fab button is used to add the passwords with name and in this home page we show all the details we have added previously in our app with the card style.)

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    app:layout_constraintBottom_toTopOf="@+id/containerLayout"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintHorizontal_bias="0.0"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintVertical_bias="0.0" />

<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:id="@+id/containerLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"

    app:layout_constraintBottom_toBottomOf="parent"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintHorizontal_bias="0.8"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintVertical_bias="1.0">

<com.google.android.material.floatingactionbutton.FloatingActionButton

    android:id="@+id/floating_action_button"
    android:layout_width="68dp"

    android:layout_height="wrap_content"
```

```

        android:layout_gravity="bottom|right"
            android:layout_margin="16dp"

        android:src="@drawable/ic_baseline_add_24"

        app:backgroundTint="@color/teal_200"
            app:fabSize="normal"
            android:clickable="true"
            android:focusable="true"
            app:rippleColor="@color/teal_700"
    />

</androidx.coordinatorlayout.widget.CoordinatorLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

[privlok/datahelper.java\(update\)](#)

```

public class
MyDataHelper
extends
SQLiteOpenHelper
{
    Context context;
    final static public String dbname="MyData";
    final static public int version=5;
    final static public String table_name="Privlok";
    final static public String col_1="id";
    final static public String col_2="title_name";
    final static public String col_3="user_name";
    final static public String col_4="account_number";
    final static public String col_5="password";
    final static public String query="create table
    "+table_name+" ( "+col_1+" integer PRIMARY KEY
    AUTOINCREMENT, "+col_2+" text, "+col_3+" text, "+col_4+"
    text, "+col_5+" text )";

    public MyDataHelper(Context context) {
        super(context, dbname, null , version);
        this.context=context;
    }

    @Override

```

```

        public void onCreate(SQLiteDatabase db) {
            db.execSQL(query);
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
            db.execSQL("drop table if exists "+table_name);
            onCreate(db);
        }
        public long insertRecord(String title_name, String
user_name, String account_number, String password)
        {
            ContentValues cv = new ContentValues();
            cv.put(col_2,title_name);
            cv.put(col_3,user_name);
            cv.put(col_4,account_number);
            cv.put(col_5,password);
            SQLiteDatabase db = this.getWritableDatabase();
            long check = db.insert(table_name, null,cv);
            return check;
        }

        Cursor readAllData(){
            String query = " Select "+col_2+" , "+col_3+" ,
"+col_4+" , "+col_5+" From " + table_name;
            SQLiteDatabase db= this.getReadableDatabase();

            Cursor cursor= null;
            if(db!=null){
                cursor=db.rawQuery(query,null);
            }
            return cursor;
        }

        void deleteOneRow(String row_id){
            SQLiteDatabase db = this.getWritableDatabase();
            long result = db.delete(table_name, col_1+"=?",
new String[]{" "+row_id});
            if(result == -1){
                Toast.makeText(context, "Failed to Delete.",
Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(context, "Successfully
Deleted.", Toast.LENGTH_SHORT).show();
            }
        }

        void deleteAllData(){
            SQLiteDatabase db = this.getWritableDatabase();
            db.execSQL("DELETE FROM " + table_name);
        }

        void updateData(String row_id, String title_name,
String user_name, String account_number, String password)

```



```

{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    //cv.put(col_1,row_id);
    cv.put(col_2,title_name);
    cv.put(col_3,user_name);
    cv.put(col_4,account_number);
    cv.put(col_5,password);

    long result = db.update(table_name, cv,
col_1+"=?", new String[]{" "+row_id});
    if(result == -1){
        Toast.makeText(context, "Failed",
Toast.LENGTH_SHORT).show();
    }else {
        Toast.makeText(context, "Updated
Successfully!", Toast.LENGTH_SHORT).show();
    }
}
}

```

[privlok/listview.xml\(update\)](#)

```

<androidx.cardview.widge
t.CardView

```

```

    android:id="@+id/cardView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp">

```

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="12dp"
    android:orientation="vertical">

```

```

    <TextView
        android:id="@+id/textViewTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="10dp"
        android:text="SBI Bank"

```

```

    android:textAppearance="@style/Base.TextAppearanc
e.AppCompat.Large" />

```

```

    <TextView

```

```

        android:id="@+id/textViewUsername"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nirmit Bansal"

        android:textAppearance="@style/Base.TextAppearance
        e.AppCompat.Medium" />

        <TextView
            android:id="@+id/textViewAccountNo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="823482748343"

            android:textAppearance="@style/Base.TextAppearance
            e.AppCompat.Medium" />

        <TextView
            android:id="@+id/textViewPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:text="nir12345" />

    </LinearLayout>

    </androidx.cardview.widget.CardView>
</LinearLayout>

```

[privlok/updateActivity.java\(update\)](#)

(In this activity we update all the previous detail that is added by the user. In update Activity we use delete button also for delete the particular activity that is stored in the form of card at the home page activity.)

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_update);
    textUpdateTitle = findViewById(R.id.text_update_title);
    textUpdatePassword =
    findViewById(R.id.text_update_password);
    textUpdateAccount =

```

```

findViewById(R.id.text_update_account);
    textUpdateUsername =
findViewById(R.id.text_update_username);
    update_button = findViewById(R.id.btnUpdate);
    delete_button = findViewById(R.id.btnDelete);

    //First we call this
    getAndSetIntentData();

    //Set actionbar title after getAndSetIntentData method
    ActionBar ab = getSupportActionBar();
    if (ab != null) {
        ab.setTitle(title);
    }

    update_button.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //And only then we call this
            MyDataHelper myDB=new
MyDataHelper(UpdateActivity.this);
            /* title =
textUpdateTitle.getEditText().getText().toString().trim();
            username =
textUpdateUsername.getEditText().getText().toString().trim();
            account =
textUpdateAccount.getEditText().getText().toString().trim();
            password =
textUpdatePassword.getEditText().getText().toString().trim();*/
            myDB.updateData(id, title, username, account,
password);
        }
    });
    delete_button.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            confirmDialog();
        }
    });

}
void getAndSetIntentData(){
    if(getIntent().hasExtra("title") &&
        getIntent().hasExtra("username") &&
getIntent().hasExtra("account") &&
        getIntent().hasExtra("password")){
        //Getting Data from Intent
        //id = getIntent().getStringExtra("id");
        title = getIntent().getStringExtra("title");
        username = getIntent().getStringExtra("username");
        account = getIntent().getStringExtra("account");
        password = getIntent().getStringExtra("password");
        //Setting Intent Data

```

```

        textUpdateTitle.getEditText().setText(title);
        textUpdateUsername.getEditText().setText(username);
        textUpdateAccount.getEditText().setText(account);
        textUpdatePassword.getEditText().setText(password);
        //Log.d("nirmit", title+" "+username+" "+account+"
"+password);
    }else{
        Toast.makeText(this, "No data.",
Toast.LENGTH_SHORT).show();
    }
}
}

```

[privlok/updateActivity.xml\(update\)](#)

```

<com.google.android.material.textfield
    .TextInputLayout

```

```

        android:id="@+id/text_update_title"

```

```

        android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
            android:layout_margin="20dp"
            app:errorEnabled="true">

```

```

<com.google.android.material.textfield
    .TextInputEditText

```

```

        android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
            android:hint="Title"
            android:textSize="30dp"

```

```

        android:background="#00000000"
            android:inputType="text"/>

```

```

</com.google.android.material.textfield
    .TextInputLayout>

```

```

<com.google.android.material.textfield
    .TextInputLayout

```

```

        android:id="@+id/text_update_username"

```

```

        android:layout_width="match_parent"

```

```
        android:layout_height="wrap_content"
            app:counterEnabled="true"
            app:counterMaxLength="20"
            android:layout_margin="20dp"
            app:errorEnabled="true">
```

```
<com.google.android.material.textfield
    .TextInputEditText
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
            android:hint="Username"
            android:textSize="20dp"
```

```
        android:background="#00000000"
            android:inputType="text"/>
```

```
</com.google.android.material.textfield
    .TextInputLayout>
```

```
<com.google.android.material.textfield
    .TextInputLayout
```

```
        android:id="@+id/text_update_account"
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
            android:layout_margin="20dp"
            app:counterEnabled="true"
            app:counterMaxLength="16"
            app:errorEnabled="true">
```

```
<com.google.android.material.textfield
    .TextInputEditText
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
            android:hint="Account
Number"
            android:textSize="20dp"
```

```
        android:background="#00000000"
```

```
        android:inputType="number"/>
```

```
</com.google.android.material.textfield
    .TextInputLayout>
```

```
<com.google.android.material.textfield
.TextInputLayout
```

```
    android:id="@+id/text_update_password"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
        android:layout_margin="20dp"
        app:errorEnabled="true"
```

```
    app:passwordToggleEnabled="true">
```

```
<com.google.android.material.textfield
.TextInputEditText
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
        android:hint="Password"
```

```
    android:background="#00000000"
        android:textSize="20dp"
```

```
    android:inputType="textPassword"/>
```

```
</com.google.android.material.textfield
.TextInputLayout>
```

```
    <Button
        android:id="@+id/btnUpdate"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
        android:text="Update"
        android:textSize="20dp"/>
```

```
    <Button
        android:id="@+id/btnDelete"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
        android:text="Delete"
        android:textSize="20dp"/>
```

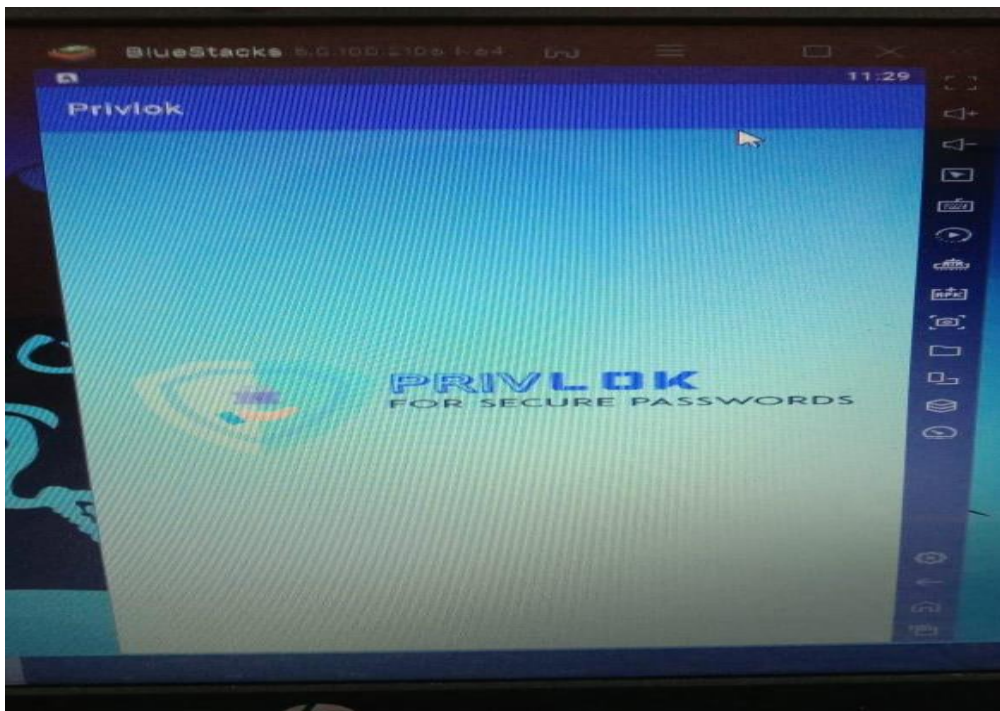
```
</LinearLayout>
```

CHAPTER 6

APPLICATION FORMAT

6.1 STEP 1

(This is our first interface of our privlok application which mention the name itself that would be privlok.)



6.2 STEP 2

(we can fill our details like title, username and password for storing into the database.)

The image shows a mobile application interface for a form titled "Privlok". The form is displayed on a screen with a blue header bar. The form fields are as follows:

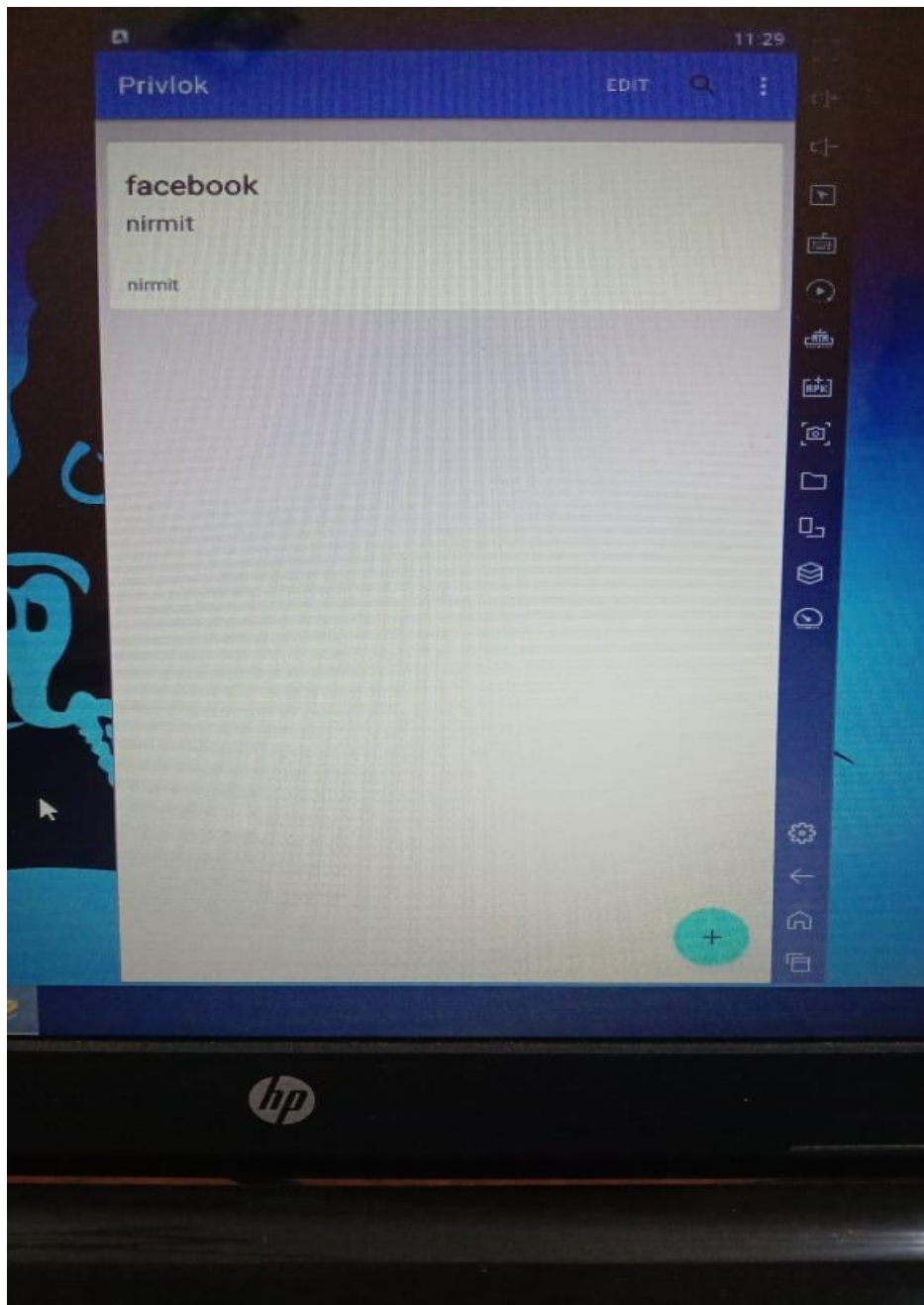
- Title**: A text input field with a blue underline.
- Username**: A text input field with a blue underline and a character count of "0/20" on the right.
- Account Number**: A text input field with a blue underline and a character count of "0/16" on the right. A mouse cursor is visible over this field.
- Password**: A text input field with a blue underline and an eye icon on the right to toggle visibility.

Below the form fields is a large blue button labeled "SAVE".

The bottom of the screen shows a black bar with the HP logo.

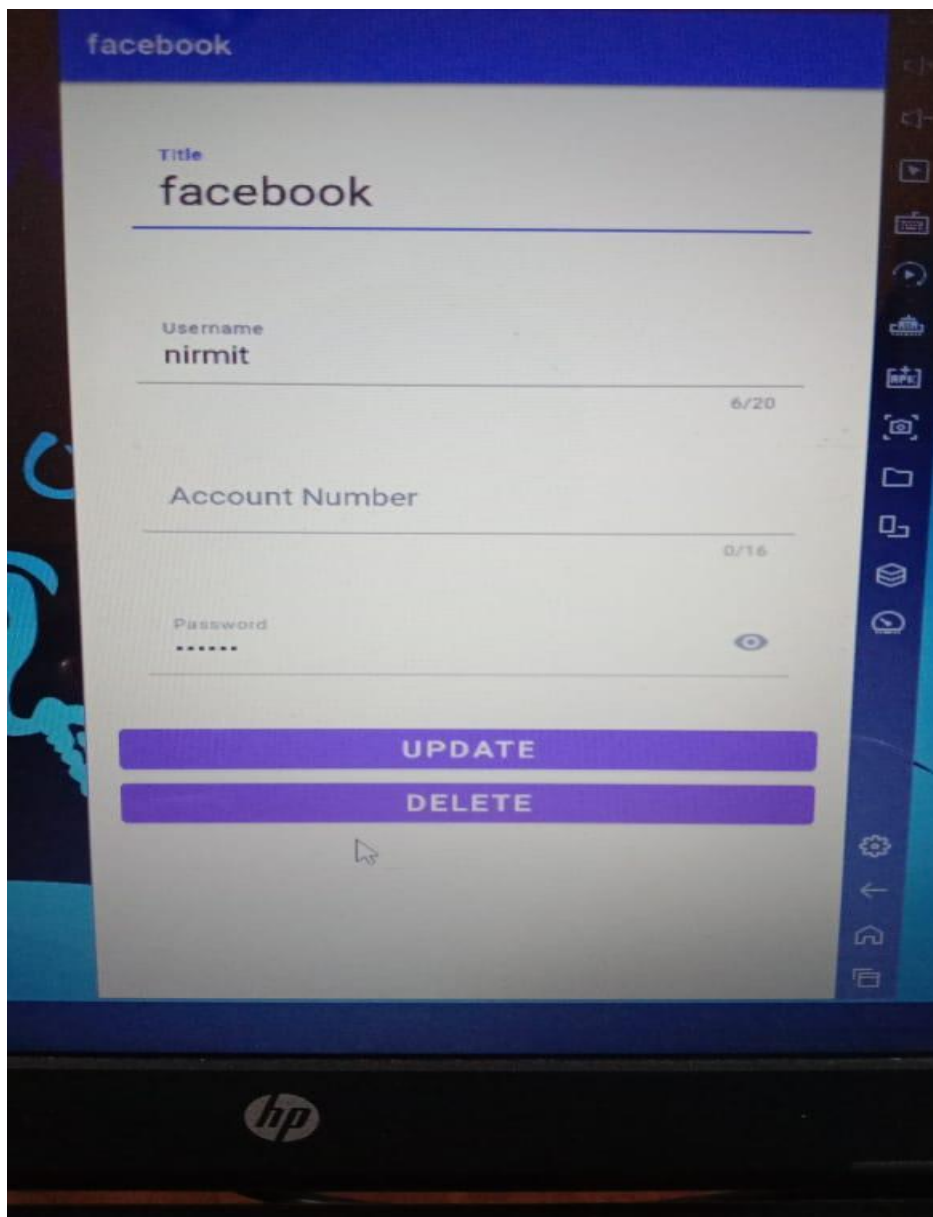
6.3 STEP 3

(Store all data)



6.4 STEP 4

(This is our edit interface in which we can fill our details like title, username and password for storing into the database.)



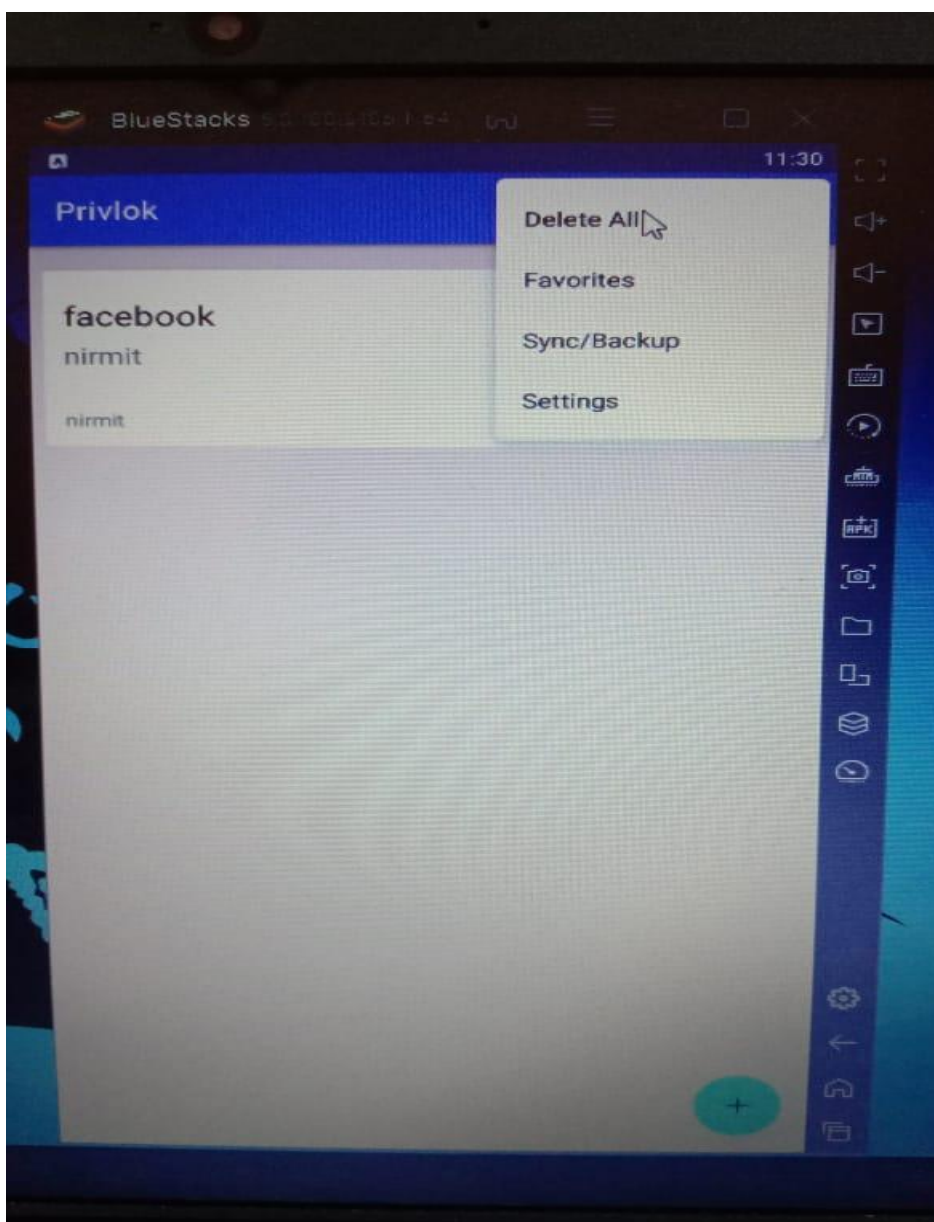
The screenshot shows a web application interface for editing a record. The title bar at the top left says "facebook". The form contains the following fields:

- Title:** The value "facebook" is entered. A character count "6/20" is visible to the right.
- Username:** The value "nirmit" is entered. A character count "6/20" is visible to the right.
- Account Number:** The field is empty. A character count "0/16" is visible to the right.
- Password:** The field contains six asterisks "*****". A toggle icon (an eye) is visible to the right.

At the bottom of the form are two large blue buttons: "UPDATE" and "DELETE". A mouse cursor is pointing at the "DELETE" button. On the right side of the screen, there is a vertical toolbar with various icons for navigation and editing. The HP logo is visible on the bottom bezel of the monitor.

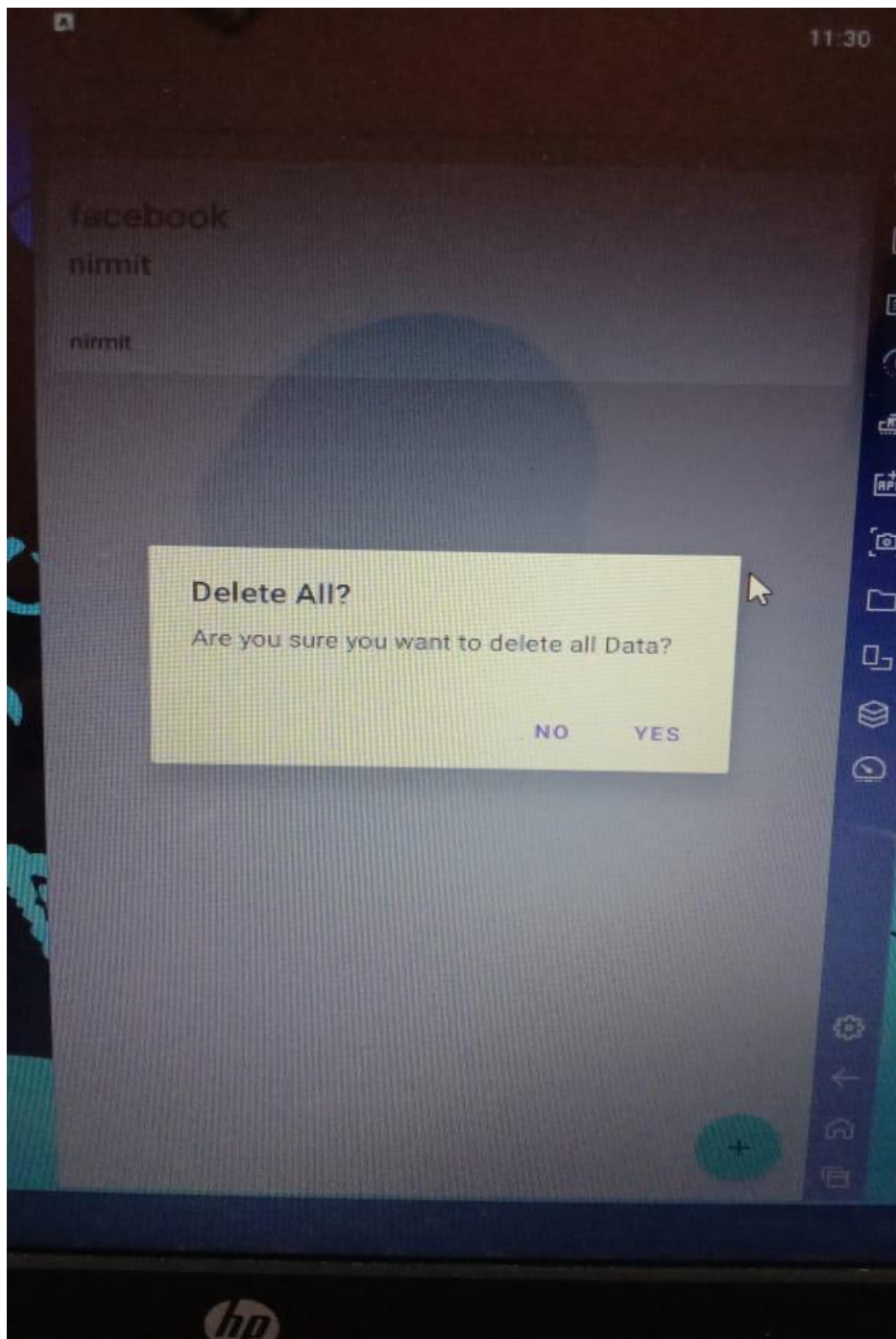
6.5 STEP 5

(If we want to delete all the stored information so we can go to the top right corner to select the delete all option.)



6.6 STEP 6

(After click on the delete all option it shows a confirmation interface to confirm either user really want to delete all the data or is it by mistake.)



CHAPTER 7

CONCLUSION

Security is the first thing that comes to our mind before using it. In day-to-day life, we have a lot of different accounts on different platforms. And the challenging task is to remember all the passwords. That is the biggest problem in today's digital world for everyone. So it is required to make a solution for it. Solution: For this problem, we have to decide to make an application which is meet the requirement of what we need. After all the research we decided to make an application (Privlok) which stored all the credential information in encrypted form for security purpose through which the new user make there an account on privlok and existing user can directly log in. Users can store their passwords with their account handle. User information will store in encrypted form. Our project is based on android technology. The technology which we are using in our application is as follows .i.e., Android Studio and SQL. The benefit of our application in the real world is that users do need to remember all the passwords all the time. Users can access their data whenever is required.

REFERENCES

1. SQLite with JDBC for Beginners: Learn Fundamentals of Queries and Implement NetBeans-Based Projects Easily by Vivian Siahaan (Author), Rismon Hasiholan Sianipar (Author)(2019).
2. SQLite For Beginners: Learn Fundamentals of Queries and Implement PyQt-Based Projects Easily by Vivian Siahaan (Author), Rismon Hasiholan Sianipar (Author)(2019).
3. SQLite Forensics by Paul Sanderson (Author), Dr. Richard Hipp (Editor), Brett Shavers (Editor), Heather Mahalik (Editor), Eric Zimmerman (Editor)(2018).
4. Getting Started with SQL: A Hands-On Approach for Beginners 1st Edition by Thomas Nield (Author)(2016).
5. SQL - The Shortest Route For Beginners: A fast and easy track for beginners that covers Oracle, MySQL, Microsoft SQL Server, Microsoft Access, IBM ... SQLite, and MariaDB SQL implementations by Riaz Ahmed (Author)(2015).
6. The Definitive Guide to SQLite Softcover reprint of the original 1st ed. Edition by Mike Owens (Author)(2015).
7. Practical Android: 14 Complete Projects on Advanced Techniques and Approaches Paperback – Import, 4 February 2018 by Mark Wickham (Author)
8. Head First Android Development: A Brain-Friendly Guide Paperback – 1 January 2017 by Dawn Griffiths (Author), David Griffiths (Author).
9. Android App Development for Dummies, 3ed Paperback – 1 January 2015 by Michael Burton (Author).
10. Learn SQLite in 1 Day: Definitive Guide to Learn SQLite for Beginners by Krishna Rungta (Author)(2017).

11. SQLite for Mobile Apps Simplified: Step by step details to create and access database from Android, BlackBerry and iPhone Apps 1.0 Edition by Sribatsa Das (Author)(2014).
12. Mastering PhoneGap Mobile Application Development by Kerri Shotts (Author)(2016).

