Chapter 1

INTRODUCTION

Manual Training and Placement which is done at various colleges is by human intervention due to which there is a maximum chance of errors. This app has been developed to override the problems prevailing the in practicing manual system. This system is supported to eliminate and in some cases reduce the hardships faced by this existing system. The aim is to automate the existing system by easy access of the data or information.

1.1 Motivation:

Training and Placement system automates activities of Training and placement cell and place the best coordination between student. It provides student community to use collective intelligence to increase selection ratio and eases out process of creation of management information automatically. Training and Placement focuses on automation of placement cell. Our project mainly helps in improving productivity and makes use of utilization of resources. There is no duplication of work as this was not the case when done manually. Thus it reduces labor and increases morale.

1.2 Proposed System:

The system is an application which will be accessed and effectively used throughout the organization with proper login enabled. It can also be used as an application for the Placement Officers in the college to manage the student information about placement thus reducing the manual work and consumes less paperwork. p- to-date information of all the students in the college. Our system also help the college to overcome the difficulty in keeping records of hundreds of students and searching for a student eligible for recruitment criteria from the whole thing. It helps in effective and timely utilization of resources. The project facilitates user friendly, reliable and fast management system. The placement officer itself can carry out operations in a smooth and effective manner. They need not concentrate on record keeping. The college can maintain computerized records thus reducing paper work, time and money.

1.3 Related Work:

The mobile applications that are available in the market are very useful to students and make their life easy.

The Placement and Training App is also one of those applications, which do not waste valuable time. As there are many similar applications available today, we added some innovative features to make our application unique, easy to use and execute.

Chapter 2

REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

2.1 Software Requirements Specification

- Android Operating System on the Smartphone.
- Android studio IDE and Android SDK
- OS: Windows 8/8.1/10/11 (64-bit)

2.2 Hardware Requirements Specification

- 8BG RAM
- 500GB Hard Disk
- I5 processor

Chapter 3

SYSTEM DESIGN

3.1 Detailed design

1.Login/Sign Up Activity:

In signup there are two fields one for email and password. The student can enter their email and password to create a new account. If the user is already registered then they can login using their registered email and password. If the email id and password is not correct then a toast message Authentication failed! is displayed.

2.Main Activity:

After successfully logging in, the user can view all the listed departments of the college. The user can choose their particular department to further view their department related activities and other placement and training related study materials.

3.Department Home Page:

There are two main activities in the department which are the study resources and the scheduler. The user can switch between these two activities by sliding.

4.Study Resources Activity

The user can then select among the various categories for placement preparation they are Aptitude, Interview preparation, Data structures and algorithms with competitive programming study materials. The links to the resources have been given for the respective categories.

5.Sheduler:

In the scheduler , the students can see what companies are coming for campus placements. This feature will help the students to plan accordingly and keeps them upto date with all the placement related activities taking place in the department. By this the students can apply and register for the companies beforehand and they can also prepare for the placement with the study resources already provided.

3.2 Application design

This application like most of the applications will have user login screen and option for registration. The user must register in this application when he/she is using for first time. However, the user who is already registered can login to the application using his/her login credentials that are created by the user at the time of registration.

3.2.1 Introduction to the Layout Editor

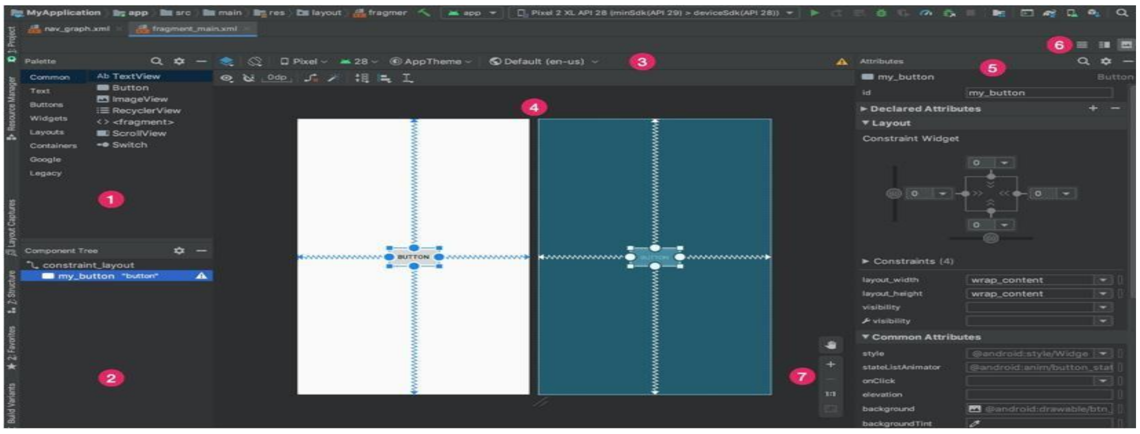


Fig No.3.2.1 Introduction of layout editor

1. Palette: Contains various views and view groups that you can drag into your layout.
2. Component Tree: Shows the hierarchy of components in your layout.
3. Toolbar: Click these buttons to configure your layout appearance in the editor and change layout attributes.
4. Design editor: Edit your layout in Design view, Blueprint view, or both.
5. Attributes: Controls for the selected view's attributes.
6. View mode: View your layout in either Code , Design , or Split modes. Split mode shows both the Code and Design windows at the same time.
7. Zoom and pan controls: Control the preview size and position within the editor.
8. When you open an XML layout file, the design editor opens by default, as shown in figure 1. To edit the layout XML in the text editor, click the Code button in the top-right corner of the window. Note that the Palette, Component Tree, and Attributes windows are not available while editing your layout in Code view.

3.2.2 Change the Preview Appearance

The buttons in the top row of the design editor enable you to configure the appearance of your layout in the editor. Corresponding to the numbers in figure 2, the buttons available are as follows:

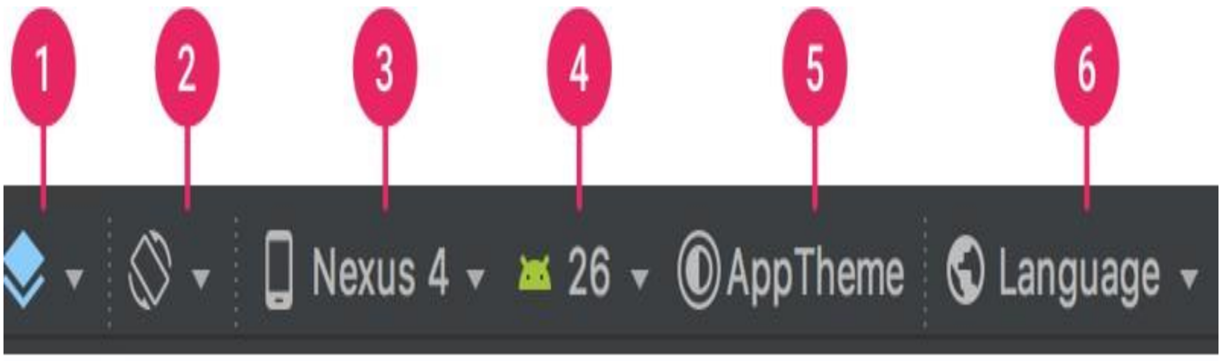


Fig 3.2.2 Buttons in the layout editor toolbar

1.Design and blueprint: Select how you'd like to view your layout in the editor. Choose Design to see a rendered preview of your layout. Choose Design + Blueprint to see both views side-by-side. You can also press B to cycle through these view types.

2.Screen orientation and layout variants: Choose between landscape and portrait screen orientation, or choose other screen modes for which your app provides alternative layouts, such as night mode. You can also press O to change orientation

3.Device type and size: Select the device type (phone/tablet, Android TV, or Wear OS) and screen configuration (size and density). You can select from several pre-configured device types and your own AVD definitions, or you can create a new AVD by selecting Add Device Definition from the list. You can resize the device size by dragging the bottom-right corner of the layout. You can also press D to cycle through the device list.

4.API version: Select the version of Android on which to preview your layout.

5.App theme: Select which UI theme to apply to the preview. Note that this works only for supported layout styles, so many themes in this list result in an error.

6.Language: Select the language to show for your UI strings. This list displays only the languages available in your string resources. If you'd like to edit your translations, click Edit Translations from the drop-down menu. For more information on working with translations, see Localize the UI with Translations Editor.

3.2.3 Create a new Layout

When adding a new layout for your app, first create a default layout file in your project's default layout/ directory so that it applies to all device configurations. Once you have a default layout, you can create layout variations for specific device configurations, such as for large screens.

You can create a new layout in one of the following ways:

Use Android Studio's main menu

1. In the Project window, click the module in which you want to add a layout.
2. In the main menu, select File > New > XML > Layout XML File.
3. In the dialog that appears, provide the file name, the root layout tag, and the source set in which the layout belongs.
4. Click Finish to create the layout.

Use the Projected View

1. Choose the Project view from within the Project window.
2. Right-click the layout directory where you'd like to add the layout.
3. In the context menu that appears, click New > Layout Resource File. Use the Android view

1. Choose the Android view from within the Project window.
2. Right-click the layout folder.
3. In the context menu that appears, select New > Layout Resource File.

Use the Resource Manager

1. In the Resource Manager, select the Layout tab.
2. Click the + button, and then click Layout Resource File.

3.3 Use Case diagram:

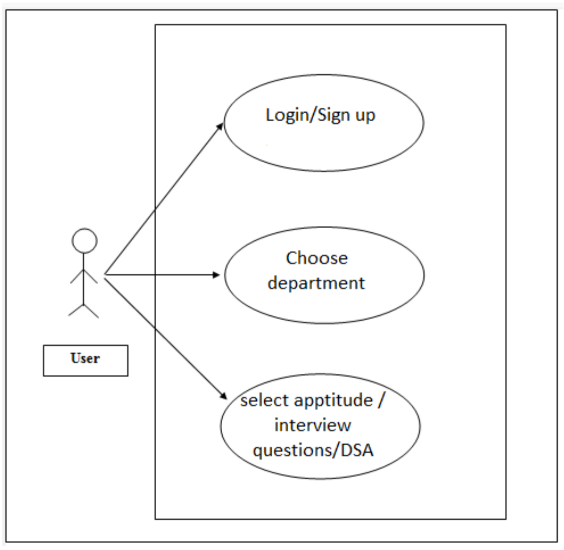


Fig 3.3.1 Case diagram

After successfully logging in, user can choose their particular department and can then select among the various categories for placement preparation.

3.4 Firebase

Authentication: This service aims to facilitate users in registering and logging in. Firebase Authentication can use Google, Twitter, Facebook, GitHub, and cell phone numbers. The system to be built uses authentication with a email id. The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with our iOS, Android and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

How does it work?

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real time events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great real time experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly

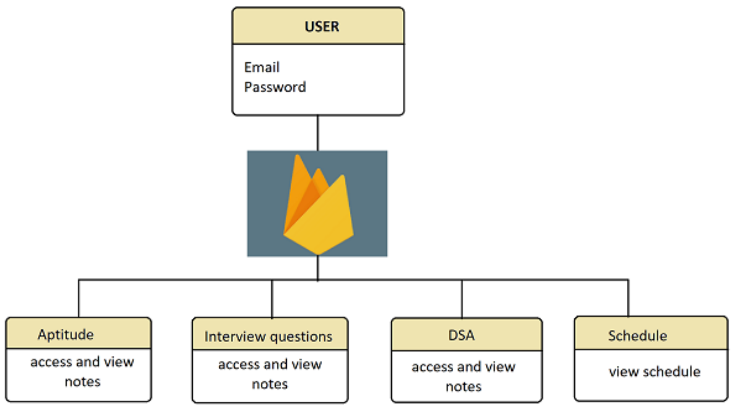


Fig 3.4 ER diagram

Chapter 4

IMPLEMENTATION AND TESTING

4.1 Introduction to Programming languages, IDES, Tools and Technologies

4.1.1Java programming languages

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc. Java Platform is a collection of programs that help programmers to develop and run Java programming applications efficiently. It includes an execution engine, a compiler, and a set of libraries in it. It is a set of computer software and specifications. James Gosling developed the Java platform at Sun Microsystems, and the Oracle Corporation later acquired it.

Here are some important Java applications:

- It is used for developing Android Apps
- Helps you to create Enterprise Software
- Wide range of Mobile java Applications
- Scientific Computing Applications

History of java programming language

Here are the important landmarks of java programming language

- The Java language was initially called OAK.
- Originally, it was developed for handling portable devices and set-top boxes. Oak was a massive failure.
- In 1995, Sun changed the name to "Java" and modified the language to take advantage of the burgeoning www (World Wide Web) development business.
- Later, in 2009, Oracle Corporation acquired Sun Microsystems and took ownership of three key Sun software assets: Java, MySQL, and Solaris.

4.1.2 Java Development Kit (JDK)

JDK is a software development environment used for making applets and Java applications. The full form of JDK is Java Development Kit. Java developers can use it on Windows, macOS, Solaris, and Linux. JDK helps them to code and run Java programs. It is possible to install more than one JDK version on the same computer.

Here are the main reasons for using JDK:

JDK contains tools required to write Java programs and JRE to execute them. It includes a compiler, Java application launcher, Applet viewer, etc.

Compiler converts code written in Java into byte code.

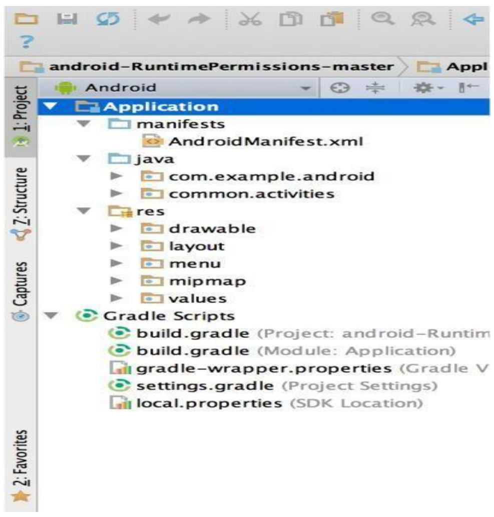
Java application launcher opens a JRE, loads the necessary class, and executes its main method.

4.1.3 Android System

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

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- Pro Guard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations[18]
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine[19]
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Project Structure



```
android-RuntimePermissions-master > Appl
Android
▼ Application
    ▼ manifests
          AndroidManifest.xml
    ▼ java
        ► com.example.android
        ► common.activities
    ▼ res
        ► drawable
        ► layout
        ► menu
        ► mipmap
        ► values
▼ Gradle Scripts
      build.gradle (Project: android-Runtim
      build.gradle (Module: Application)
      gradle-wrapper.properties (Gradle V
      settings.gradle (Project Settings)
      local.properties (SDK Location)
```

Fig 4.1.3.1 The Project Files in Android View

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

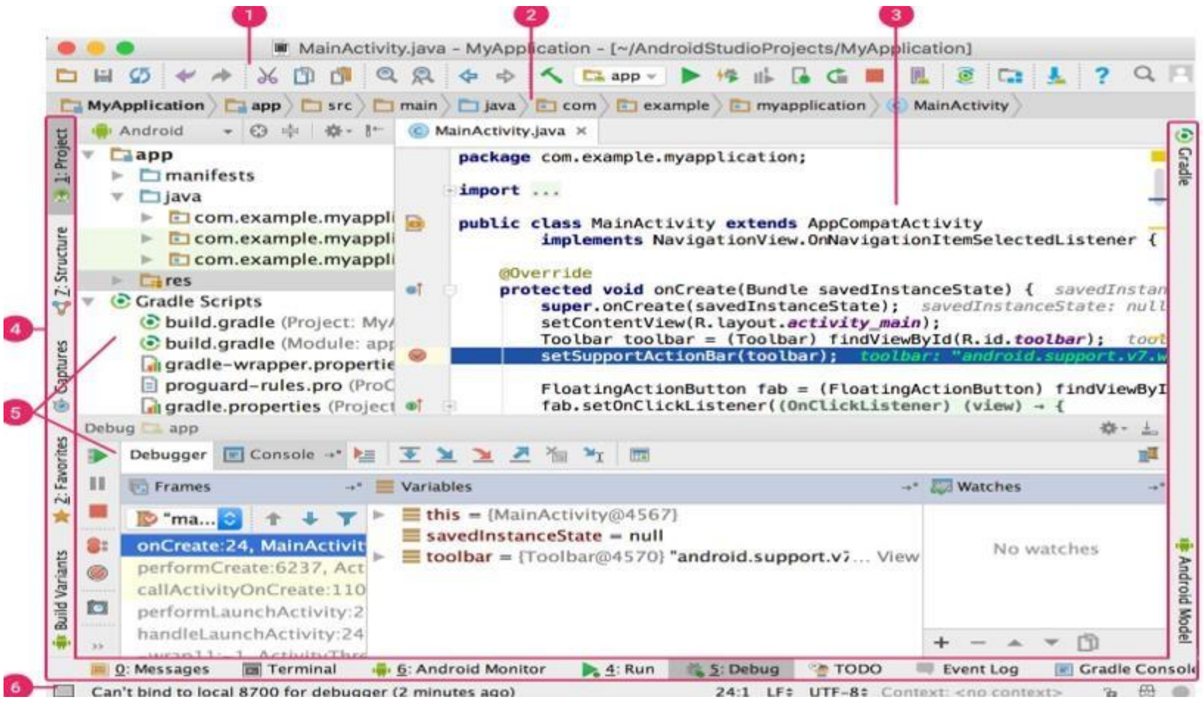
All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select Project from the Project dropdown (in figure 1, it's showing as Android).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the Problems view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

The User Interface



```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {  savedInstan
        super.onCreate(savedInstanceState);  savedInstanceState: null
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  tool
        setSupportActionBar(toolbar);  toolbar: "android.support.v7.w

        FloatingActionButton fab = (FloatingActionButton) findViewByI
        fab.setOnClickListener((OnClickListener) (view) → {
```

```
"ma...
onCreate:24, MainActivit
performCreate:6237, Act
callActivityOnCreate:110
performLaunchActivity:2
handleLaunchActivity:24
```

```
this = {MainActivity@4567}
savedInstanceState = null
toolbar = {Toolbar@4570} "android.support.v7... View
```

```
Can't bind to local 8700 for debugger (2 minutes ago)        24:1  LF÷  UTF-8÷  Context: <no context>
```

Fig 4.1.3.2 The Android Studio Main Window

The Android Studio main window is made up of several logical areas identified in figure:

1.The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

2.The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

3.The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4.The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5.The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6.The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

7.The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

8.The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

9.The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

10.The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

11.The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

12.The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Gradle Build System

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plugin for Gradle. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

Customize, configure, and extend the build process.

Create multiple APKs for your app, with different features using the same project and modules. Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core

source files. Android Studio build files are named build gradle. They are plain text files that use Groovy syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

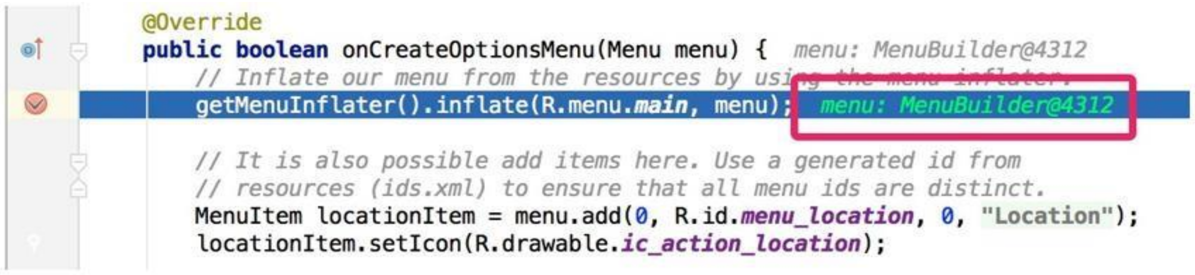
Debug and profile tools

Android Studio assists you in debugging and improving the performance of your code, including inline debugging and performance analysis tools.

Inline Debugging

Use inline debugging to enhance your code walk-throughs in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

- Inline variable values
- Referring objects that reference a selected object
- Method return values
- Lambda and operator expressions
- Tooltip values



```
@Override
public boolean onCreateOptionsMenu(Menu menu) {  menu: MenuBuilder@4312
    // Inflate our menu from the resources by using the menu inflater.
    getMenuInflater().inflate(R.menu.main, menu);  menu: MenuBuilder@4312

    // It is also possible add items here. Use a generated id from
    // resources (ids.xml) to ensure that all menu ids are distinct.
    MenuItem locationItem = menu.add(0, R.id.menu_location, 0, "Location");
    locationItem.setIcon(R.drawable.ic_action_location);
```

Fig 4.1.3.3 An inline variable value

To enable inline debugging, in the Debug window, click Settings and select the checkbox for Show **Values Inline.**

Performance Profiles

Android Studio provides performance profilers so you can more easily track your app’s memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests.

With your app running on a device or emulator, open the **Android Profiler tab**. Annotations in Android Studio

Android Studio supports annotations for variables, parameters, and return values to help you catch bugs, such as null pointer exceptions and resource type conflicts. The Android SDK Manager packages the Support- Annotations library in the Android Support Repository for use with Android Studio. Android Studio validates the configured annotations during code inspection.

4.2 Test Plan and Test Activities

Android Studio is designed to make testing simple. With just a few clicks, you can set up a JUnit test that runs on the local JVM or an instrumented test that runs on a device. Of course, you can also extend your test capabilities by integrating test frameworks such as Mockito to test Android API calls in your local unit tests, and Espresso or UI Automator to exercise user interaction in your instrumented tests. You can generate Espresso tests automatically using Espresso Test Recorder.

Test types and location

The location of your test code depends on the type of test you are writing. Android Studio provides source code directories (source sets), for the following two types of tests:

Local unit tests

Located at module-name/src/test/java/.

```
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals("com.smallacademy.userroles", appContext.getPackageName());
    }
}
```

These are tests that run on your machine's local Java Virtual Machine (JVM). Use these tests to minimize execution time when your tests have no Android framework dependencies or when you can mock the Android framework dependencies.

At runtime, these tests are executed against a modified version of android.jar where all final modifiers have been stripped off. This lets you use popular mocking libraries, like Mockito.

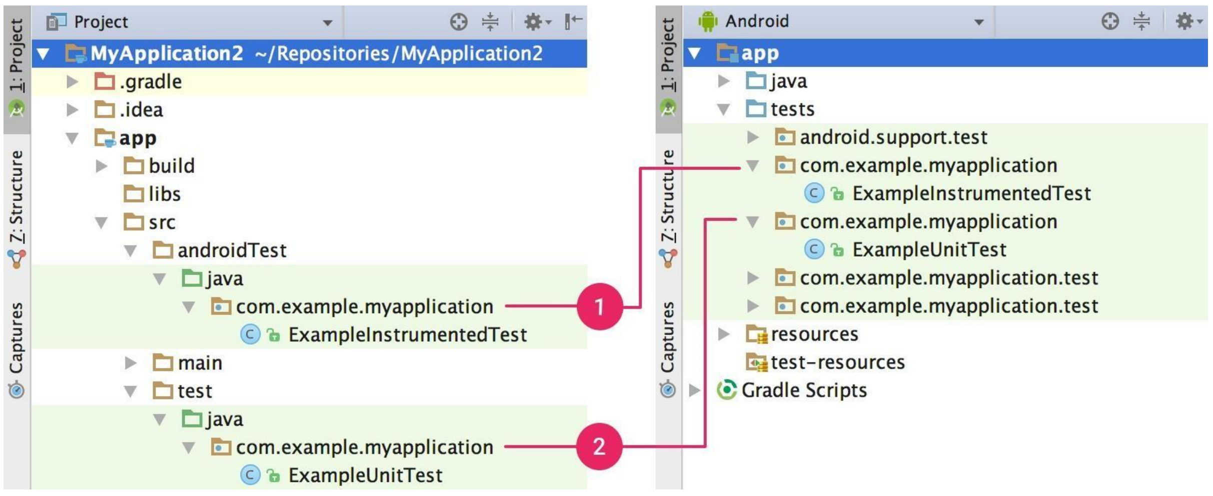


Figure4.2.1. Your project's (1) instrumented tests and (2) local JVM tests are visible in either the Project view (left)or Android view (right)

Add a new test

To create either a local unit test or an instrumented test, you can create a new test for a specific class or method by following these steps:

1. Open the Java file containing the code you want to test.

2. Click the class or method you want to test, then press Ctrl+Shift+T (⇧⌘T).

3. In the menu that appears, click Create New Test.

4. In the Create Test dialog, edit any fields and select any methods to generate, and then click OK.

5. In the Choose Destination Directory dialog, click the source set corresponding to the type of test you want to create: androidTest for an instrumented test or test for a local unit test. Then click OK.

Alternatively, you can create a generic Java file in the appropriate test source set as follows:

1.In the Project window on the left, click the drop-down menu and select the Project view.

2.Expand the appropriate module folder and the nested src folder. To add a local unit test, expand the test folder and the nested java folder; to add an instrumented test, expand the androidTest folder and the nested java folder.

3.Right-click on the Java package directory and select New > Java Class.

4.Name the file and then click OK. Run a test

To run a test, proceed as follows:

- Be sure your project is synchronized with Gradle by clicking Sync Project in the toolbar.
- Run your test in one of the following ways:
- In the Project window, right-click a test and click Run .
- To run all tests, right-click on the test directory and click Run tests .
- In the Code Editor, right-click a class or method in the test file and click Run to test all methods in the class.

Xml Code:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView 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=".MainActivity"
    android:background="#FAAD8C">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <androidx.cardview.widget.CardView
            android:id="@+id/c1"
            android:layout_width="match_parent"
            android:layout_height="230dp"
            android:layout_margin="10dp"
            android:background="@color/navy">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:src="@drawable/cse_final"
                android:background="@color/navy"/>
        </androidx.cardview.widget.CardView>
        <androidx.cardview.widget.CardView
            android:id="@+id/c2"
            android:layout_below="@+id/c1"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:layout_margin="10dp"
            android:background="@color/white">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:src="@drawable/ece_final"
                android:background="@color/navy"/>
        </androidx.cardview.widget.CardView>
        <androidx.cardview.widget.CardView
            android:id="@+id/c3"
            android:layout_below="@+id/c2"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:layout_margin="10dp"
            android:background="@color/white">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:src="@drawable/ecee_final"
                android:background="@color/navy"/>
        </androidx.cardview.widget.CardView>
        <androidx.cardview.widget.CardView
            android:id="@+id/c4"
            android:layout_below="@+id/c3"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:layout_margin="10dp"
            android:background="@color/white">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:src="@drawable/civil_final"
                android:background="@color/navy"/>
        </androidx.cardview.widget.CardView>
        <androidx.cardview.widget.CardView
            android:id="@+id/c5"
            android:layout_below="@+id/c4"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:layout_margin="10dp"
            android:background="@color/white">
            <ImageView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:src="@drawable/mech_final"
                android:background="@color/navy"/>
        </androidx.cardview.widget.CardView>
    </RelativeLayout>
</ScrollView>
```

Java Code:

```
package com.example.placement;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.util.Patterns;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class signIn extends AppCompatActivity {

    private EditText mEmail,mPass ;
    private TextView mTextView;
    private Button signInbtn;
    private FirebaseAuth mAuth;

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

        mEmail=findViewById(R.id.email_reg);
        mPass=findViewById(R.id.pass_reg);
        mTextView=findViewById(R.id.textView2);
        signInbtn=findViewById(R.id.signin_btn);
        mAuth=FirebaseAuth.getInstance();

        mTextView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(signIn.this,MainActivity.class));
            }
        });
        signInbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loginUser();
            }
        });
    }
    private void loginUser(){
        String email=mEmail.getText().toString();
        String pass=mPass.getText().toString();

        if(!email.isEmpty()&&
Patterns.EMAIL_ADDRESS.matcher(email).matches()){
            if (!pass.isEmpty()){
                mAuth.signInWithEmailAndPassword(email,pass).addOnSuccessLi
stener(new OnSuccessListener<AuthResult>() {
                    @Override
                    public void onSuccess(AuthResult authResult) {
                        Toast.makeText(signIn.this, "Login Successful",
Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(signIn.this ,
home.class));
                        finish();
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(signIn.this, "Login Failed !!",
Toast.LENGTH_SHORT).show();
                    }
                });

            }
            else{
                mPass.setError("Empty fields not allowed");
            }

        }
        else if (email.isEmpty()){
            mEmail.setError("Empty fields are not allowed");

        }
        else{
            mEmail.setError("Please enter Correct Email");
        }

    }
}
```

```
public class CseActivity extends AppCompatActivity

{

    TabLayout tabLayout;

    TabItem tabItem1,tabItem2;

    ViewPager2 viewPager;



    PageAdapter pageAdapter;



    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_cse);



        tabLayout = (TabLayout) findViewById(R.id.tablayout1);
        viewPager = (ViewPager2) findViewById(R.id.vpager);

//          tabLayout.addTab(tabLayout.newTab().setText("Resources"));
//          tabLayout.addTab(tabLayout.newTab().setText("Scheduler"));

        FragmentManager  fragmentManager
=getSupportFragmentManager();
        pageAdapter = new PageAdapter(fragmentManager ,
getLifecycle());

        viewPager.setAdapter(pageAdapter);

        tabLayout.addOnTabSelectedListener(new
TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {

                viewPager.setCurrentItem(tab.getPosition());

            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }
        });
        viewPager.registerOnPageChangeCallback(new
ViewPager2.OnPageChangeCallback() {

            @Override

            public void onPageSelected(int position) {

                tabLayout.selectTab(tabLayout.getTabAt(position));

            }

        });

      }

    }
```

Chapter 5

RESULTS AND DISCUSSIONS

5.1 Screenshots

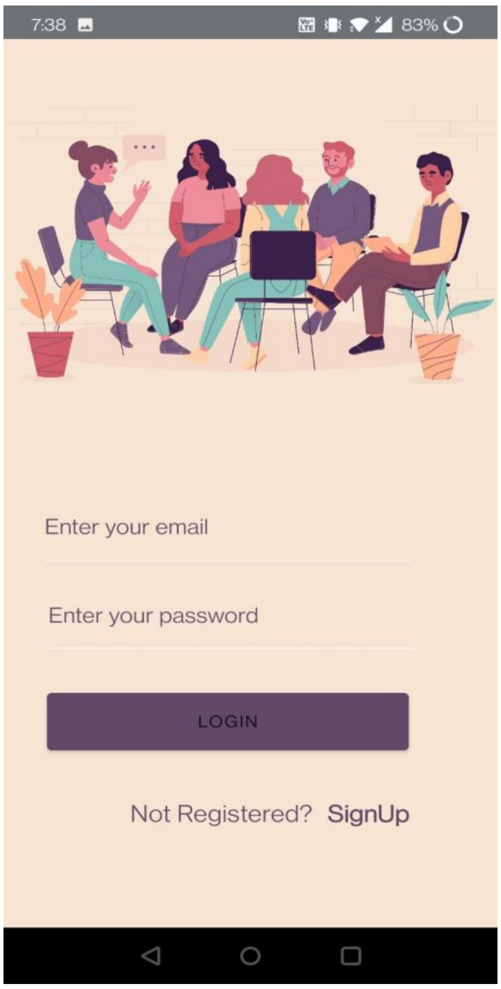


Fig 5.1.1 Login Page

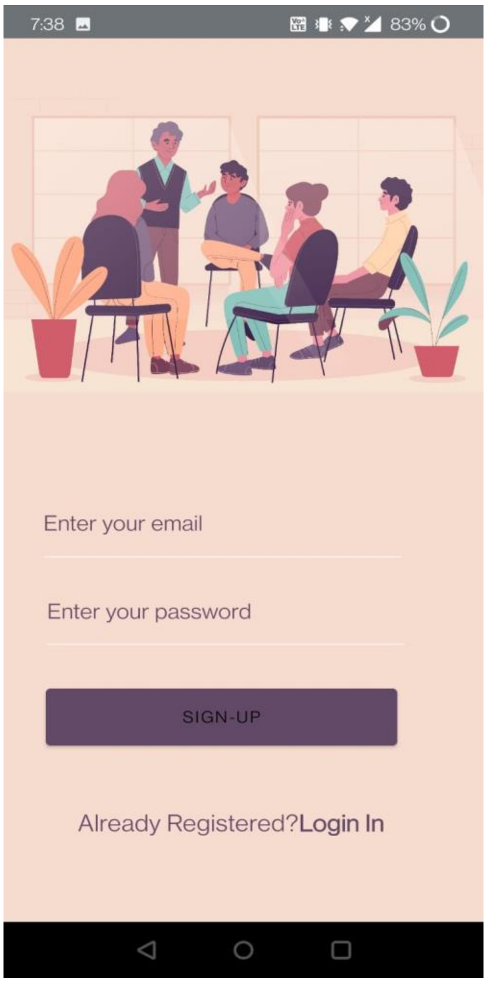


Fig 5.1.2 Sign Up Page

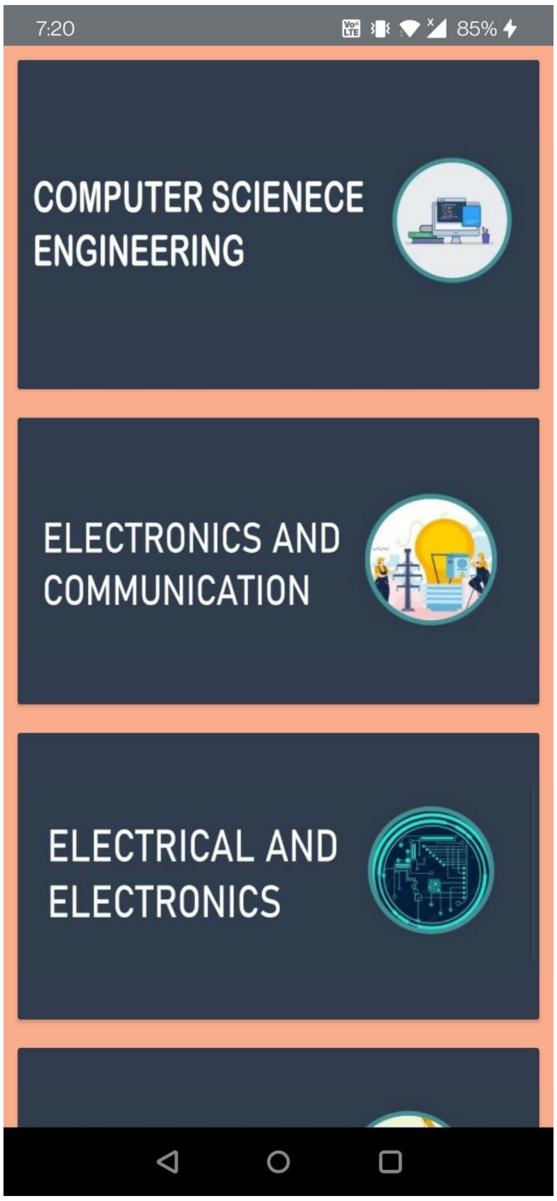

7:20
85%
COMPUTER SCIENECE ENGINEERING
ELECTRONICS AND COMMUNICATION
ELECTRICAL AND ELECTRONICS

Fig 5.1.3 list of departments

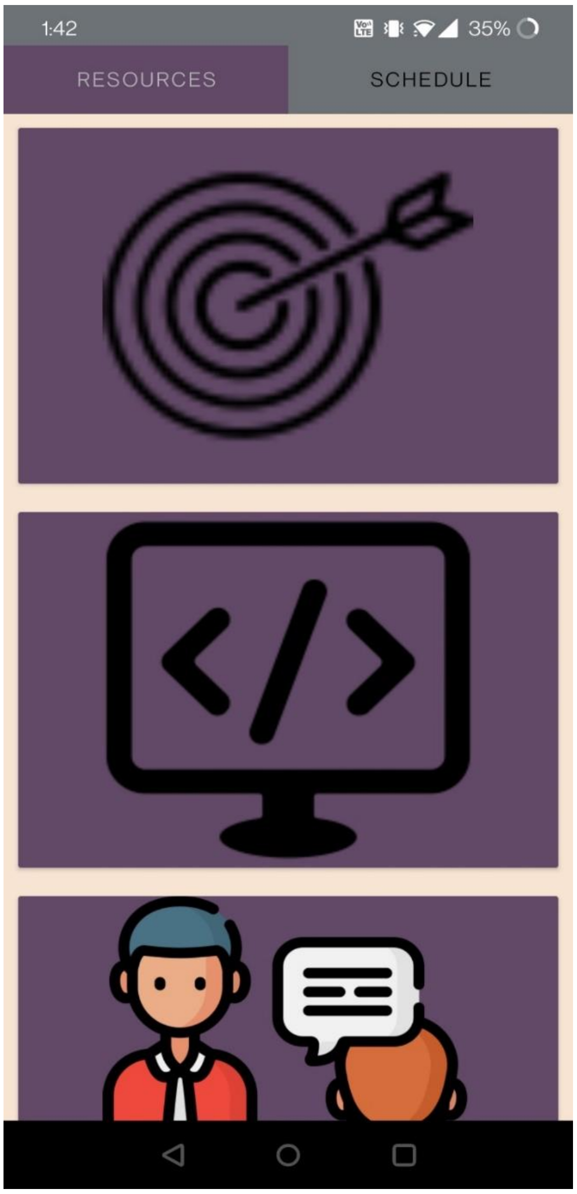
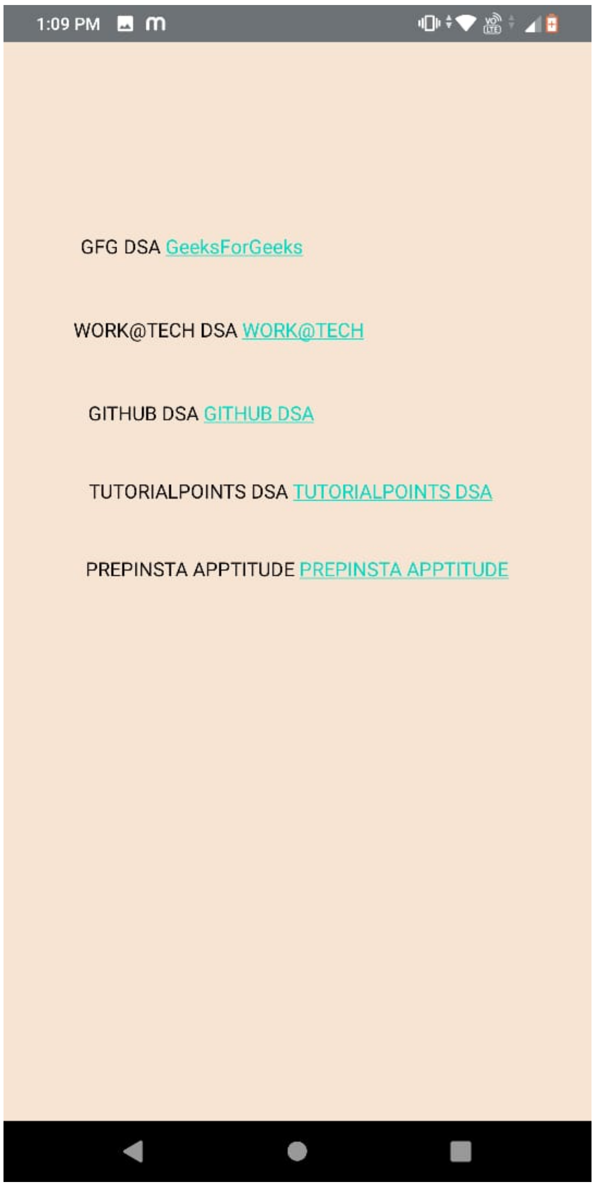


Fig 5.1.4 Study Resource


1:09 PM

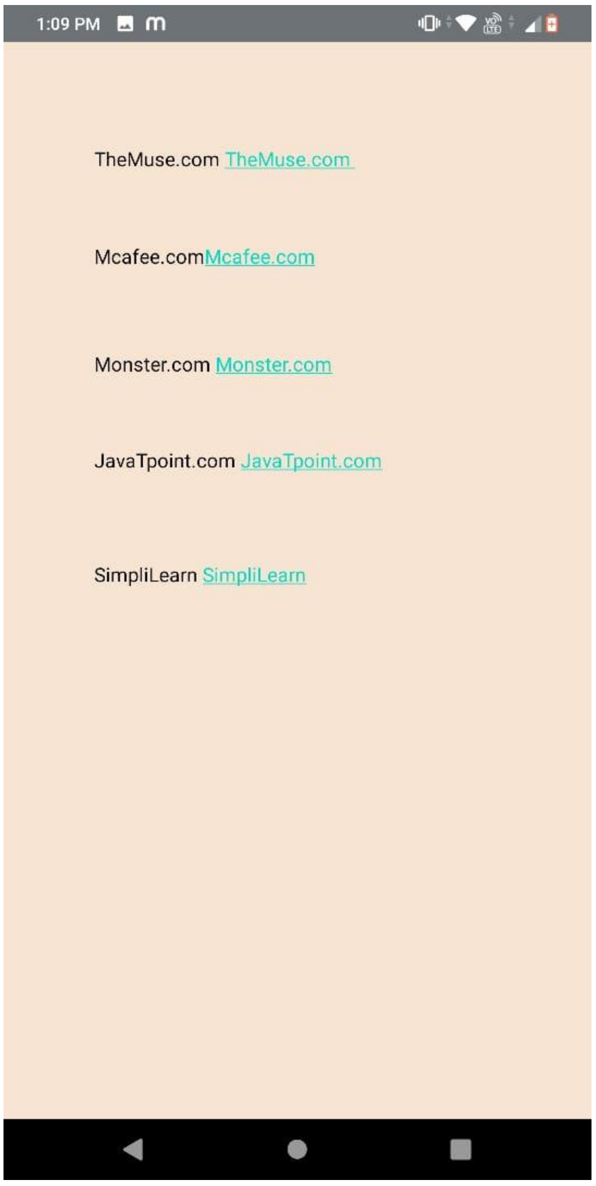
GFG DSA GeeksForGeeks

WORK@TECH DSA WORK@TECH

GITHUB DSA GITHUB DSA

TUTORIALPOINTS DSA TUTORIALPOINTS DSA

PREPINSTA APPTITUDE PREPINSTA APPTITUDE


1:09 PM

TheMuse.com TheMuse.com

Mcafee.comMcafee.com

Monster.com Monster.com

JavaTpoint.com JavaTpoint.com

SimpliLearn SimpliLearn

Fig 5.1.4 Study Resources -Aptitude Fig 5.1.5 list of departments

FUTURE WORK

- In future there is a chance to generate graphs on placement procedure on the bases of database.
- In future there is a scope for staff/coordinators add their study materials and clarify.
- In future we can add an alert domain for the sake of students.
- In future we can add a Feedback from student to faculty.
- In future we can add Company information through company page links.
- In future we can add Chatbot for clarifying our doubts

CONCLUSION

Maximum work goes manually in the present placement system which makes it take time to avail changes. This includes main problems like searching for the data of students and sorting them along with it. Also, updating student data is a cumbersome job and does not have a method to notify the student in time which makes the management of the placements very difficult. In the proposed system, all of these problems become automated. The registration of the student for an upcoming placement, the addition of a new user, notifying students, sharing information is all met.

BIBLIOGRAPHY

[1] Rajnesh Tripathi, Raghvendra Singh, Jaweria Usmani, ”Campus Recruitment and Placement Sys- tem”, International Conference on Recent Innovations in Science and Engineering(Icrise-18), April, 2018.

[2] https://www.youtube.com/c/MdJamalmca

[3] https://www.youtube.com/c/AndroidDevelopers

[4] https://developer.android.com/guide/components/fundamentals

[5] https://www.youtube.com/c/firebase