Project Report On

“Project Management System**”**

Submitted by

**Wani Ashwin Balasaheb Walunje Darshan Dilip**

**Ritesh Sandip Somwanshi Shinde Vaman Mahadu**

Under Guidance of Prof. A.R.Mirakar



Prof. V. A. Parajane

**Department of computer technology**

**Sanjivani Rural Education Society’s**

**Sanjivani k. B. P. Polytechnic Kopargaon - 423603 AHMEDNAGAR 2020-2021**

## DEPARTMENT OF COMPUTER TECHNOLOGY

**Sanjivani Rural Education Society’s**

**SANJIVANI K. B. P. POLYTECHNIC**



CERTIFICATE

## This is the certify that the Project report entitled

**‘Project Management System’**

Submitted by

**Wani Ashwin Balasaheb Walunje Darshan Dilip**

**Somwanshi Ritesh Sandip Shinde Vaman Mahadu**

Under our supervision and guidance for partial fulfillment of the requirement for

Diploma in Computer Technology affiliated to Maharashtra State Board of Technical Education, Mumbai

Prof. V. A. Parajane …….……………………..

Project Coordinator

And project guide External Examiner

Prof. A.R.Mirikar Principal

**ACKNOWLEDGEMENT**

We feel great pleasure of submitting the project report on “Academic Web Portal with Android Application”. Every orientation work has imprint of many people and this work is no different.

This work gives us an opportunity to deep for the same. While preparing project report. We received endless help from number of people. This report would be incomplete if we don’t convey my sincere thanks to all those who were involved.

We are highly thankful for the cooperation extended by **Prof.A.R. Mirikar** principal, Sanjivani K.B.P. Polytechnic, Kopargaon for giving comments for improvement of work, encouragement and help during completion of project.

We would like to personally thank **Prof.G.N. Jorvekar** Sir (Head of Computer Dept.) for providing us all the necessary facilities for the successful completion of the project.

**Mr. Wani Ashwin Balasaheb**

**Mr.Walunje Darshan Dilip**

**Mr. Somwanshi Ritesh Sandip**

**Mr. Shinde Vaman mahadu**

**Diploma in Computer Technology Sanjivani K. B. P. Polytechnic, Kopargaon**

**Abstract**

Project Management System, to make communication easy between Project co- ordinator(PCC) and student.

As this application can help Project co-ordinator (PCC) as well as student.

Student have to install this application, in their smart phone, and open and account to use the features of this application

With this application manual work go on digital form, as student can upload documents, view group guide info, notification etc. Instead of wasting paper and time too.

For Project co-ordinator (PCC) this software is so playing a main role.

.

As this application user-friendly, so anyone can easily operate it. As our main motive is providing services of particular institute.

To maintain the data of particular group, taking paper work on digital platform is the main highlight of this application

# Contents ACKNOWLEDGEMENT3 ABSTRACT4

## INTRODUCTION

* 1. INRODUCTION 8
  2. [PROBLEM STATEMENT 9](#_TOC_250020)
  3. LITERATURE SURVEY 10

1. REQUIREMENT ANALYSIS
   1. [REQUIREMENT SPECIFICATION 11](#_TOC_250019)
      1. [Normal Requirements 11](#_TOC_250018)
      2. [Expected Requirements 11](#_TOC_250017)
      3. [Excited Requirements 11](#_TOC_250016)

[2.2 SYSTEM REQUIREMENT..................](#_TOC_250015)

1. Hardware Requirement 12
2. Software Requirement 12
3. SYSTEM ANALYSIS
   1. [PROCESS MODEL 13](#_TOC_250014)
      1. [Spiral Model 16](#_TOC_250013)
      2. [When to Use Spiral Methodology? 17](#_TOC_250012)
   2. [SYSTEM BREAKDOWN ANALYSIS STRUCTURE 18](#_TOC_250011)
4. SYSTEM BREAKDOWN STRUCTURE 19
5. SYSTEM BLOCK DIAGRAM/FLOW DIAGRAM 20
   1. [DATA-FLOW DIAGRAMS 21](#_TOC_250010)
   2. [USE-CASE DIAGRAM 23](#_TOC_250009)
   3. [ACTIVITY DIAGRAM 24](#_TOC_250008)
6. IMPLEMENTATION DETAILS
   1. [INTRODUCTION 25](#_TOC_250007)
   2. [DATABASE USED: 26](#_TOC_250006)

4.4 MAJOR CODE 27

1. TESTING, RESULTS AND ANALYSIS
   1. INRODUCTION 35
      1. [White-Box Testing 35](#_TOC_250005)
      2. Black Box Testing 36
      3. Unit Testing 36
      4. Integration Testing 36
   2. [TESTCASES 37](#_TOC_250004)
   3. [VALIDATION OF TESTING: .................](#_TOC_250003)
      1. User Registration Form 38
      2. User Login Form: 38
      3. Dashboard Activity: 39

5.3.4 Upload Documents Page: . . . . . . . . . . . . . 39. . . . . . .

* + 1. Student Login (Android App) 40
    2. Main Activity: 40

5.3.7 Note Viewer Activity: . . . . . . . . . . . . . . . . . .41 . . .

5.3.8 Feedback Activity: 41

1. ADVANTAGES AND FUTURE SCOPE
   1. [ADVANTAGES 42](#_TOC_250002)
   2. [FUTURE SCOPE 42](#_TOC_250001)
2. CONCLUSION AND REFERENCES
   1. [CONCLUSION 43](#_TOC_250000)
   2. REFERENCES 43

List of Figures

* 1. Scenario of Academic Web Portal with Android App . . . . . . . . . . . .

3.1 Spiral Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

* 1. System Breakdown Structure Analysis . . . . . . . . . . . . . . .
  2. System Breakdown Structure . . . . . . . . . . . . . . . . . . . .

3.4 System Block Diagram. . . . . . . . . . . . . . . . . . . .

3.5 Use Case diagram . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.6 Activity Diagram . . . . . . . . . . . . . . . . . . . . . . . . . . .

5.1 Registration form . . . . . . . . . . . . . . . . . . . . . . . . . . .

5.2 Login Form . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

5.3 Dashboard Form . . . . . . . . . . . . . . . . . . . . . . .

5.4 Upload Documents Form . . . . . . . . . . . . . . . . . . . . . . . . . .

5.5 Student Login (Android App) . . . . . . . . . . . . . . . . . . . . . . . . . . . .

5.6 Main Activity . . . . . . . . . . . . . . . . . . . . . . . . . .

Chapter: 1

Chapter 1.Introduction

* We design Project Management System, to make communication easy between Project coordinator (PCC) and student.
* Its user friendly software, so that is why they do not require specially trained person to handle the software.
* Project Management System, provides a detailed view of how the students project details.
* The Project management system also equipped with some special features for student.
* Each record in the system has unique identity, and it can be searched by a unique id.
* Use of this project in the campus reduce the paper work .
* As many work like Group guide information, upload documents, notifications, view document, view group details.
* This application can provide PCC to get personal attention on groups activities .
* Secure data of groups and PCC can be maintain, as all the complex data will be easier to find particular data just a click away.
* As all campus need an efficient work as our application can make so easy just a click away things to be done
* As its frontend of this application uses of Android studio and XML (Android toll kit) with Java as Backend.

# Problem statement

* + - Generally all kind of project work is done on manual paper work, as manual maintenance.
    - Student project details is done on manual paper work.
    - The main motive is help out in solving the problem of communication between PCC and student about the project.
    - The main use of this app is to manage all projects and their details.
  1. **Literature Survey**
     + Project Management System, provides a detailed view of how the students project details.
     + The Project management system also equipped with some special features for student.
     + Each record in the system has unique identity, and it can be searched by a unique id.
     + Use of this project in the campus reduce the paper work .
     + As many work like Group guide information, upload documents, notifications, view document, view group details.
     + This application can provide PCC to get personal attention on groups activities .
     + Secure data of groups and PCC can be maintain, as all the complex data will be easier to find particular data just a click away.
     + As all campus need an efficient work as our application can make so easy just a click away things to be done
* As its frontend of this application uses of Android studio and XML (Android toll kit) with Java as Backend

Chapter 2

**REQUIREMENT ANALYSIS**

Requirement analysis is a software engineering task that bridges the gap between system level software description and design model.

The system description describes overall system functionality of the system including software, hardware, databases, human interfaces and other system elements.

The software design mainly focuses on application architectural, user interface and component level designs.

As per problem definition and scope of project discussed in the previous chapter, the requirement analysis from the point of software and hardware has been performed. The requirements have been elaborated in following sections.

* 1. **REQUIREMENT SPECIFICATION**

The first work was to gather information on various modules like implementation of common password and analysis of various algorithms.

The information about this was collected from various references .

We downloaded various PDF files about various algorithms. We have taken the advice and suggestions form the principle, staff about the project and its implementation.

### Normal Requirements

These are the requirements clearly stated by the customer hence these requirements must be present for customer satisfaction.

N1: The system must add the PCC, students.

N2: The documents must get uploaded by the PCC and student at the application. N3: Android platform must login student.

N4: Uploaded data must be seen at the android platform.

### Expected Requirements

These are the requirements clearly stated by the customer hence these requirements must be present for customer satisfaction.

E1: Keep all the data/info of the student. E2: System should reduce the paper work. E3: Smart campus must be initiated.

### Excited Requirements

These requirements are implicit type of requirements. These requirements are not clearly stated by the customer but implicitly comes during system design.

X1: System should be user friendly.

X2: System should be platform independent. X3: ERP system included.

### SYSTEM REQUIREMENT

* + 1. **Hardware Requirement**
       1. **ANDROID PLATFORM**
          - Processor: Pentium core i3 onwards
          - RAM: 4GB (Min)
          - HDD: 500GB (Min)
    2. **Software Requirement**
* Software used:-Android studio
* For storing the data:- (Firebase cloud)
  + Front End….

Xml (Android studio design tool)

* + Back End….

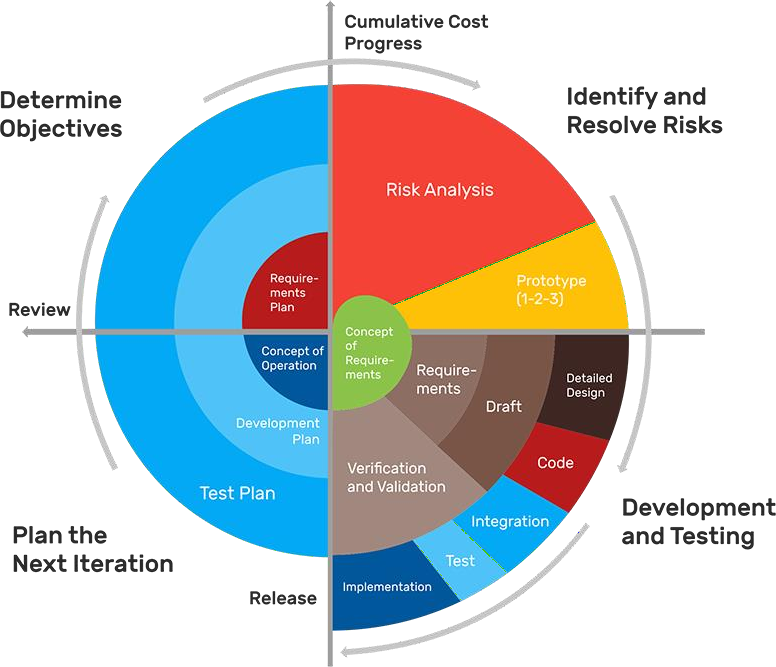
Java

**Chapter 3**

**SYSTEM ANALYSIS**

### PROCESS MODEL

Spiral model is used as the process model in our system. Fig 3.1 shows the process model of the system.



**Figure 3.1: Spiral Model**

### Spiral Model

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model. The Radius of the spiral at any point represents the expenses (cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

1. **Communication-**

Software development process starts with the communication between customer and developer. According to need of project, we gather the requirement related to project.

1. **Planning-**

It includes complete estimation and project scheduling and tracking, also includes estimation of project cost and time.

1. **Modelling-**

Task requires building of one or more representation of the application. It includes analysis and design. It is a multiple process that includes four attributes of program data structure, software architecture, interface representation and procedural details.

1. **Construction-**

Construction incrementally fills in the architecture with production ready code produce from analysis, design, and implementation with testing of the function requirements.

1. **Deployment-**

Include delivery of the partially completed/implemented project and taken feedbacks. The feedback is considered while reconstruction of the project

### When to Use Spiral Methodology?

* + - 1. When project is large
      2. When releases are required to be frequent
      3. When creation of a prototype is applicable
      4. When risk and costs evaluation is important
      5. For medium to high-risk projects
      6. When requirements are unclear and complex
      7. When changes may require at any time
      8. When long term project commitment is not feasible due to changes in economic priorities

### SYSTEM BREAKDOWN ANALYSIS STRUCTURE

Android based Project Management System

T4:2:

Design

T4:1:

Analysis

T4: Modeling

T1:2:

Requirement

T1:

Project initiation

T1:

communication

T2:2:

Project scheduling an and

Tracking

T2:1:

Estimation

T2: Planning

T3: Risk

Figure 3.2: System Breakdown Structure Analysis

* 1. **SYSTEM BREAKDOWN STRUCTURE**

Breakdown structure of the Academic web portal with android application shown below. In this the main system is divided into three parts:



Project-coordintor

Student

Registration



Admin(pcc)

Dashboard

Android based Project Management System



Notification

View Group



Upload Document

Guide Contact

View Document

View Document

Figure 3.2: System Breakdown Structure Analysis



Personal Notification

Upload Notification

* 1. **SYSTEM BLOCK DIAGRAM/FLOW DIAGRAM**



Group 3

Group-Guide

Group 2

PCC

Group 1

Figure 3.4: Internal Architecture of the System

### DATA FLOW DIAGRAMS

Data Flow Diagrams can be used to provide the end user with a physical idea of where the data is input ultimately has an effect upon the structure of the whole system from where to order and where to dispatch to report.

Student

Firebase

Figure 3.5: Level 0 DFD

Project

Management System

Firebase

PCC

### USE CASE DIAGRAM

Use Case diagrams are a set of use cases, actors and their relationships.

They represent the Use case view of a system. A use case represents a particular functionality of a system.

Fire

base

View Group

View Document

Notification

Guide

contact

Upload

Document

Personal

Notification

Student

Figure 3.8: Use Case Diagram

### ACTIVITY DIAGRAM

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent or branched.



Registration

Notice

Login

View Group

Dashboard

View

Document

PCC

Notification

Data base

Guide

contact

Log out

Personal

Notification

Upload

Document



PCC



Register student





Figure 3.9: Activity Diagram

Chapter 4 IMPLIEMENTATION DETAILS

* 1. **INTRODUCTION**
     + We design Project Management System, to provide best an easy interaction between student and Project co-ordinator (PCC)

Its user friendly software, so that is why they do not require specially trained person to handle the software.

Project Management System, provides a detailed view of how the students project details. The Project management system also equipped with some special features for student.

Each record in the system has unique identity, and it can be searched by a unique id. Use of this project in the campus reduce the paper work .

As many work like Group guide information, upload documents, notifications, view document, view group details.

This application can provide PCC to get personal attention on groups activities .

Secure data of groups and PCC can be maintain, as all the complex data will be easier to find particular data just a click away.

As all campus need an efficient work as our application can make so easy just a click away things to be done

Data will be store at real time database as data cannot be stolen, easier to find an execute particular operation.

Interaction between student and PCC will be more efficient. Which is important for any institute.

In the other hand our application will also protect the environment as paper work will be reduce at large numbers in campus work.

### 4.2. DATABASE USED:

Firebase is a powerful Cloud Storage, simple, and cost-effective object storage service.

The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality. Can use SDKs to store images, audio, video, or other user-generated content.

Cloud Storage for Firebase lets you upload and share user generated content, such as images and video, which allows you to build rich media content into your apps.

Data is stored in a Google cloud bucket an Exabyte scale object storage solution with high availability and global redundancy.

Cloud Storage for Firebase lets securely upload these files directly from mobile devices and web browsers, handling spotty networks with ease.

The Firebase Real-time 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 iOS, Android, and JavaScript SDKs, all of your clients share one Real-time Database instance and automatically receive updates with the newest data.

**MAJOR CODE**

* + 1. **Android Application (Java Code**)
       - Signin :

❖

* + - * package com.example.projectmanagementsystem;

❖

* + - * import androidx.annotation.NonNull;
      * import androidx.appcompat.app.AlertDialog;
      * import androidx.appcompat.app.AppCompatActivity;

❖

* + - * import android.content.Context;
      * import android.content.DialogInterface;
      * import android.content.Intent;
      * import android.content.SharedPreferences;
      * import android.net.ConnectivityManager;
      * import android.net.NetworkInfo;
      * import android.os.Build;
      * import android.os.Bundle;
      * import android.preference.PreferenceManager;
      * import android.provider.Settings;
      * import android.view.View;
      * import android.widget.Button;
      * import android.widget.CheckBox;
      * import android.widget.ProgressBar;
      * import android.widget.Toast;

❖

* + - * import com.google.android.material.textfield.TextInputLayout;
      * import com.google.firebase.database.DataSnapshot;
      * import com.google.firebase.database.DatabaseError;
      * import com.google.firebase.database.DatabaseReference;
      * import com.google.firebase.database.FirebaseDatabase;
      * import com.google.firebase.database.Query;
      * import com.google.firebase.database.ValueEventListener;
      * public class signin extends AppCompatActivity {
      * TextInputLayout xid, xpassword;
      * Button sub;
      * CheckBox remember;
      * private ProgressBar spinner;
      * FirebaseDatabase rootNode;
      * SharedPreferences sharedpreferences;
      * public static String MyPREFERENCES = "MyPrefs";
      * String sid="";
      * DatabaseReference reference;
      * @Override
      * protected void onCreate(Bundle savedInstanceState) {
      * super.onCreate(savedInstanceState);
      * setContentView(R.layout.activity\_signin);
      * xpassword = findViewById(R.id.password);
      * xid = findViewById(R.id.gid);
      * remember = findViewById(R.id.remember);
      * sub=findViewById(R.id.submit) ;
      * spinner =findViewById(R.id.progressBar1);
      * }
      * public void signindone(View v) {
      * spinner.setVisibility(View.VISIBLE);
      * sub.setVisibility(View.GONE);
      * spinner.requestFocus();
      * String upassword = xpassword.getEditText().getText().toString();
      * String uid = xid.getEditText().getText().toString().toUpperCase();
      * if(! isConnected(signin.this)){
      * showcustomalert();
      * }
      * if (!validateid() | !validatepassword()) {
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * return;
      * }
      * rootNode = FirebaseDatabase.getInstance();
      * reference = rootNode.getReference("signin");
      * Query checkuser = reference.orderByChild("id").equalTo(uid);
      * checkuser.addListenerForSingleValueEvent(new ValueEventListener() {
      * @Override
      * public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
      * if (dataSnapshot.exists()) {
      * String Dpassword =

dataSnapshot.child(uid).child("password").getValue().toString().trim();

* + - * String Did =

dataSnapshot.child(uid).child("id").getValue().toString().trim();

* + - * String Dstatus =

dataSnapshot.child(uid).child("status").getValue().toString().trim();

* + - * if (Dpassword.equals(upassword)) {
      * if (Dstatus.equals("0")) {
      * xpassword.getEditText().setText("");

❖

Toast.makeText(signin.this,"STUDENT",Toast.LENGTH\_LONG).show();

* + - * Intent i = new Intent(signin.this, studentdashboard.class);
      * Did=Did.toUpperCase();
      * i.putExtra("name",Did);
      * sid=Did;
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * session();
      * startActivity(i);
      * } else if (Dstatus.equals("1")) {
      * xpassword.getEditText().setText("");
      * Intent i = new Intent(signin.this, teacherdashboard.class);
      * Did=Did.toUpperCase();
      * i.putExtra("name",Did);
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * session();

❖

Toast.makeText(signin.this,"Teacher",Toast.LENGTH\_LONG).show();

* + - * startActivity(i);
      * }
      * } else {
      * passworderror();
      * }
      * } else {
      * usererror();
      * }
      * }
      * @Override
      * public void onCancelled(@NonNull DatabaseError databaseError) {
      * }
      * });
      * }
      * private boolean isConnected(signin signin) {
      * ConnectivityManager connectivityManager= (ConnectivityManager) signin.getSystemService(Context.CONNECTIVITY\_SERVICE);
      * NetworkInfo wificonn=connectivityManager.getNetworkInfo(ConnectivityManager.TYPE\_WIFI);
      * NetworkInfo mobileconn=connectivityManager.getNetworkInfo(ConnectivityManager.TYPE\_MOBILE);
      * if((wificonn !=null && wificonn.isConnected()) || (mobileconn !=null && mobileconn.isConnected()))
      * {
      * return true;
      * }
      * else {
      * return false;
      * }
      * }
      * private void showcustomalert() {
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * AlertDialog.Builder builder=new AlertDialog.Builder(signin.this);
      * builder.setMessage("Please connect to a network")
      * .setCancelable(false)
      * .setPositiveButton("Connect", new DialogInterface.OnClickListener() {
      * @Override
      * public void onClick(DialogInterface dialog, int which) {
      * startActivity(new Intent(Settings.ACTION\_WIFI\_SETTINGS));
      * }
      * })
      * .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
      * @Override
      * public void onClick(DialogInterface dialog, int which) {
      * startActivity(new Intent(getApplicationContext(),signin.class));
      * finish();
      * }
      * }).show();
      * }
      * public Boolean validateid(){
      * String nameval = xid.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xid.setError("Field cannot be empty");
      * return false;
      * }
      * else {
      * xid.setError(null);
      * return true;
      * }
      * }
      * public Boolean validatepassword(){
      * String nameval = xpassword.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xpassword.setError("Field cannot be empty");
      * return false;
      * } else if(nameval.length() < 6){
      * xpassword.setError("should be greater than 6 digits");
      * return false;
      * }
      * else {
      * xpassword.setError(null);
      * return true;
      * }
      * }
      * public void usererror(){
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * Toast.makeText(this,"Please check the Group ID",Toast.LENGTH\_LONG).show();
      * xid.getEditText().setText("");
      * xid.setError("Please enter correct Group ID");
      * xpassword.getEditText().setText("");
      * xid.getEditText().setText("");
      * }
      * public void passworderror(){
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * Toast.makeText(this,"Please check the password",Toast.LENGTH\_LONG).show();
      * xpassword.getEditText().setText("");
      * xpassword.setError("Please enter correct Password");
      * }
      * public void next(View v){
      * Intent i=new Intent(signin.this,signup.class);
      * startActivity(i);
      * }
      * private void session() {

❖

* + - * sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE);
      * SharedPreferences.Editor editor = sharedpreferences.edit();
      * editor.putString("good", sid);
      * editor.commit();
      * }
      * }

❖

❖

## Signup

* + - * package com.example.projectmanagementsystem;
      * import androidx.appcompat.app.AlertDialog;
      * import androidx.appcompat.app.AppCompatActivity;

❖

* + - * import android.content.Context;
      * import android.content.DialogInterface;
      * import android.content.Intent;
      * import android.net.ConnectivityManager;
      * import android.net.NetworkInfo;
      * import android.os.Bundle;
      * import android.provider.Settings;
      * import android.view.View;
      * import android.widget.Button;
      * import android.widget.ProgressBar;
      * import android.widget.Toast;

❖

* + - * import com.google.android.material.textfield.TextInputLayout;
      * import com.google.firebase.database.DataSnapshot;
      * import com.google.firebase.database.DatabaseError;
      * import com.google.firebase.database.DatabaseReference;
      * import com.google.firebase.database.FirebaseDatabase;
      * import com.google.firebase.database.ValueEventListener;
      * public class signup extends AppCompatActivity {
      * TextInputLayout xid,xname1,xname2,xname3,xname4,xclass,xemail,xphone,xpassword,xpname;
      * FirebaseDatabase rootNode;
      * private ProgressBar spinner;
      * DatabaseReference reference;
      * String emailPattern = "[a-zA-Z0-9.\_-]+@[a-z]+\\.+[a-z]+";
      * String MobilePattern = "[0-9]{10}";
      * String classpattern="[a-zA-Z]+";
      * String namepattern = "[a-zA-Z ]+";
      * String idpattern="[tbTB0-9.-]+";
      * final String status="0";
      * Button sub;
      * @Override
      * protected void onCreate(Bundle savedInstanceState) {
      * super.onCreate(savedInstanceState);
      * setContentView(R.layout.activity\_signup);

❖

* + - * if(! isConnected(signup.this)){
      * showcustomalert();
      * }
      * xid=findViewById(R.id.gid);
      * xname1=findViewById(R.id.name1);
      * xname2=findViewById(R.id.name2);
      * xname3=findViewById(R.id.name3);
      * spinner=findViewById(R.id.progressBar1);
      * sub=findViewById(R.id.go);
      * xname4=findViewById(R.id.name4);
      * xclass=findViewById(R.id.classroom);
      * xemail=findViewById(R.id.email);
      * xphone=findViewById(R.id.phone);
      * xpassword=findViewById(R.id.password);
      * xpname=findViewById(R.id.pname);
      * }
      * private boolean isConnected(signup signup) {
      * ConnectivityManager connectivityManager= (ConnectivityManager) signup.getSystemService(Context.CONNECTIVITY\_SERVICE);
      * NetworkInfo wificonn=connectivityManager.getNetworkInfo(ConnectivityManager.TYPE\_WIFI);
      * NetworkInfo mobileconn=connectivityManager.getNetworkInfo(ConnectivityManager.TYPE\_MOBILE);
      * if((wificonn !=null && wificonn.isConnected()) || (mobileconn !=null && mobileconn.isConnected()))
      * {
      * return true;
      * }
      * else {
      * return false;
      * }
      * }
      * private void showcustomalert() {
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * AlertDialog.Builder builder=new AlertDialog.Builder(signup.this);
      * builder.setMessage("Please connect to a network")
      * .setCancelable(false)
      * .setPositiveButton("Connect", new DialogInterface.OnClickListener() {
      * @Override
      * public void onClick(DialogInterface dialog, int which) {
      * startActivity(new Intent(Settings.ACTION\_WIFI\_SETTINGS));
      * }
      * })
      * .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
      * @Override
      * public void onClick(DialogInterface dialog, int which) {
      * startActivity(new Intent(getApplicationContext(),signup.class));
      * finish();
      * }
      * }).show();
      * }
      * public void go(View v){
      * spinner.setVisibility(View.VISIBLE);

❖

* + - * sub.setVisibility(View.GONE);
      * if(! isConnected(signup.this)){
      * showcustomalert();
      * }
      * if(!validatename() | !validateid() | !validateemail() | !validatepassword() |

!validatephone() | !validateclass() | !validatepname())

* + - * {
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * return;
      * }
      * DatabaseReference ref = FirebaseDatabase.getInstance().getReference();

❖

ref.child("signin").child(xid.getEditText().getText().toString()).addListenerForSingleValu eEvent(new ValueEventListener() {

* + - * @Override
      * public void onDataChange(DataSnapshot dataSnapshot) {
      * if (dataSnapshot.exists()) {
      * idExist();
      * } else {
      * rootNode = FirebaseDatabase.getInstance();
      * reference = rootNode.getReference("signin");
      * String id,name1,name2,name3,name4, password, phone, email,classr,pname;
      * id = xid.getEditText().getText().toString().toUpperCase();
      * name1 = xname1.getEditText().getText().toString();
      * name2 = xname2.getEditText().getText().toString();
      * name3 = xname3.getEditText().getText().toString();
      * name4 = xname4.getEditText().getText().toString();
      * password = xpassword.getEditText().getText().toString();
      * classr = xclass.getEditText().getText().toString();
      * phone = xphone.getEditText().getText().toString();
      * email = xemail.getEditText().getText().toString();
      * pname = xpname.getEditText().getText().toString();
      * Userhelperclass userhelperclass = new Userhelperclass(id,name1,name2,name3,name4, email, phone, password, classr, status,pname);
      * reference.child(id).setValue(userhelperclass);
      * done();
      * }
      * }
      * @Override
      * public void onCancelled(DatabaseError databaseError) {
      * }
      * });
      * }
      * public void done(){
      * Toast.makeText(this,"Registration Done Successfully",Toast.LENGTH\_LONG).show();
      * finish();
      * Intent i=new Intent(signup.this,signin.class);
      * spinner.setVisibility(View.GONE);
      * sub.setVisibility(View.VISIBLE);
      * startActivity(i);
      * }
      * private void idExist() {
      * Toast.makeText(this,"ID ALREADY PRESENT !!",Toast.LENGTH\_LONG).show();
      * xid.setError("ID Already Present");
      * xphone.getEditText().setText("");
      * xname1.getEditText().setText("");
      * xname2.getEditText().setText("");
      * xname3.getEditText().setText("");
      * xname4.getEditText().setText("");
      * xclass.getEditText().setText("");
      * xpassword.getEditText().setText("");
      * xemail.getEditText().setText("");
      * xpname.getEditText().setText("");
      * }
      * public Boolean validateid(){
      * String nameval = xid.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xid.setError("Field cannot be empty");
      * return false;
      * } else if(!nameval.matches(idpattern)){
      * xid.setError("Invalid Format");
      * return false;
      * }
      * else {
      * xid.setError(null);
      * return true;
      * }
      * }
      * public Boolean validatename(){
      * String nameval = xname1.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xname1.setError("Field cannot be empty");
      * return false;
      * } else if(!nameval.matches(namepattern)){
      * xname1.setError("Invalid Format");
      * return false;
      * }
      * else {
      * xname1.setError(null);
      * return true;
      * }
      * }
      * public Boolean validatepname(){
      * String nameval = xpname.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xpname.setError("Field cannot be empty");
      * return false;
      * }
      * else {
      * xpname.setError(null);
      * return true;
      * }
      * }
      * public Boolean validateemail(){
      * String nameval = xemail.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xemail.setError("Field cannot be empty");
      * return false;
      * } else if(!nameval.matches(emailPattern)){
      * xemail.setError("Invalid Format");
      * return false;
      * }
      * else {
      * xemail.setError(null);
      * return true;
      * }
      * }
      * public Boolean validatephone(){
      * String nameval = xphone.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xphone.setError("Field cannot be empty");
      * return false;
      * } else if(!nameval.matches(MobilePattern)){
      * xphone.setError("Invalid Format");
      * return false;
      * }
      * else {
      * xphone.setError(null);
      * return true;
      * }
      * }
      * public Boolean validateclass(){
      * String nameval = xclass.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xclass.setError("Field cannot be empty");
      * return false;
      * } else if(!nameval.matches(classpattern)){
      * xclass.setError("Invalid Format");
      * return false;
      * }
      * else {
      * xclass.setError(null);
      * return true;
      * }
      * }
      * public Boolean validatepassword(){
      * String nameval = xpassword.getEditText().getText().toString();
      * if(nameval.isEmpty()){
      * xpassword.setError("Field cannot be empty");
      * return false;
      * } else if(nameval.length() < 6){
      * xpassword.setError("should be greater than 6 digits");
      * return false;
      * }
      * else {
      * xpassword.setError(null);
      * return true;
      * }
      * }
      * public void back(View v){
      * Intent i=new Intent(signup.this,signin.class);
      * startActivity(i);
      * }

❖

❖

❖

* + - * }

## Student Dashboard

package com.example.projectmanagementsystem; import androidx.appcompat.app.AppCompatActivity;

import android.content.Context; import android.content.Intent;

import android.content.SharedPreferences; import android.os.Bundle;

import android.preference.PreferenceManager; import android.view.View;

import android.widget.Button; import android.widget.TextView;

public class studentdashboard extends AppCompatActivity { TextView user;

SharedPreferences sharedpreferences;

public static String MyPREFERENCES = "MyPrefs"; String userr="";

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_studentdashboard);

sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE); userr = sharedpreferences.getString("good", "");

String str=userr; user=(TextView)findViewById(R.id.user); Intent intent = getIntent();

str = intent.getStringExtra("name"); user.append(userr.trim());

}

public void notification(View view) {

Intent i =new Intent(studentdashboard.this,viewnotice.class); i.putExtra("name",userr);

startActivity(i);

}

public void documentview(View view) {

Intent i =new Intent(studentdashboard.this,uploadpdf.class); i.putExtra("name",userr);

startActivity(i);

}

public void guideinformation(View view) {

Intent i =new Intent(studentdashboard.this,mentor.class); i.putExtra("name",userr);

startActivity(i);

}

public void pnotification(View view) {

Intent i =new Intent(studentdashboard.this,mynotices.class); i.putExtra("name",userr);

startActivity(i);

}

}

## Teacher Dashboard:

* + - * package com.example.projectmanagementsystem;

❖

* + - * import androidx.appcompat.app.AppCompatActivity;

❖

* + - * import android.content.Intent;
      * import android.os.Bundle;
      * import android.view.View;
      * import android.widget.TextView;

❖

* + - * public class teacherdashboard extends AppCompatActivity {

❖

* + - * TextView user;
      * public String userr=" ";
      * @Override
      * protected void onCreate(Bundle savedInstanceState) {
      * super.onCreate(savedInstanceState);
      * setContentView(R.layout.activity\_teacherdashboard);

❖

* + - * user=(TextView)findViewById(R.id.user);
      * Intent intent = getIntent();
      * String str = intent.getStringExtra("name");
      * user.append(str);
      * }

❖

* + - * public void notification(View view) {
      * Intent i =new Intent(teacherdashboard.this,addnotice.class);
      * i.putExtra("name",userr);
      * startActivity(i);
      * }

❖

* + - * public void viewgroup(View view) {
      * Intent in =new Intent(teacherdashboard.this,viewgroups2.class);
      * in.putExtra("name",userr);
      * startActivity(in);

❖

* + - * }

❖

* + - * public void viewdocument(View view) {
      * Intent i =new Intent(teacherdashboard.this,viewgroups.class);
      * i.putExtra("name",userr);
      * startActivity(i);
      * }

❖

* + - * public void groupinfo(View view) {
      * Intent i=new Intent(teacherdashboard.this,viewgroups2.class);
      * startActivity(i);
      * }
      * }

## FireBase Cloud Storage:

package com.example.projectmanagementsystem;

import android.content.Context; import android.content.Intent; import android.graphics.Color; import android.provider.Settings; import android.view.LayoutInflater; import android.view.View;

import android.view.ViewGroup; import android.widget.Button; import android.widget.TextView; import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import com.firebase.ui.database.FirebaseRecyclerAdapter; import com.firebase.ui.database.FirebaseRecyclerOptions; import com.google.android.gms.tasks.OnSuccessListener; import com.google.firebase.database.DatabaseReference; import com.google.firebase.database.FirebaseDatabase; import com.google.firebase.database.collection.LLRBNode;

import java.util.HashMap;

import static androidx.core.content.ContextCompat.startActivity;

public class myfirebaseadapter extends FirebaseRecyclerAdapter<model,myfirebaseadapter.myviewholder> { public myfirebaseadapter(@NonNull FirebaseRecyclerOptions<model> options) {

super(options);

}

FirebaseDatabase rootNode; Context context; DatabaseReference reference; @Override

protected void onBindViewHolder(@NonNull myviewholder holder, int position, @NonNull model model)

{

String s=model.getStatus(); if(s.equals("0")){ holder.btn.setText(model.getId());} else{

holder.btn.setVisibility(View.GONE);

}

}

@NonNull @Override

public myviewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) { View view =

LayoutInflater.from(parent.getContext()).inflate(R.layout.activity\_singlerowdata,parent,false); return new myviewholder(view);

}

class myviewholder extends RecyclerView.ViewHolder

{

String idname="0"; Button btn;

public myviewholder(@NonNull View itemView) { super(itemView);

context = itemView.getContext();

btn=(Button) itemView.findViewById(R.id.btn1); btn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) { idname=btn.getText().toString().trim();

Intent intent = new Intent(context,RetrievePDF.class); intent.putExtra("name",idname); context.startActivity(intent);

}

});

}

}

}

## Chapter 5

### TESTING, RESULTS AND ANALYSIS

* 1. **INTRODUCTION**

Software Testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risk of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing can be stated as the process of validating and verifying that a computer program/application/product.

### White-Box Testing

White-box testing also called as clear box testing, glass box testing, transparent box testing and structural testing, tests internal structures or working of the program, as opposed to the functionality exposed to the end user, In white box testing an internal perspective of the system, as well as Programming skills, are used to design test cases. The tester chooses input to exercise path through the code and determine the appropriate output. This is analogous to testing nodes in a circuit. Though this method of test design can uncover many errors or problems.

Techniques uses in white-box testing include:

* + - 1. API Testing
      2. Code Coverage
      3. Fault Injection methods
      4. Mutation Testing methods
      5. Static Testing methods.

1. **Black Box Testing**

Black box testing treats the software as a black box, examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification- based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements.

1. **Unit Testing**

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the functional level. In an object-oriented environment, this if usually at the class level and the minimum unit tests include the constructors and destructor. This type of testing is usually written by developers

as they work on code. Unit testing along cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each.

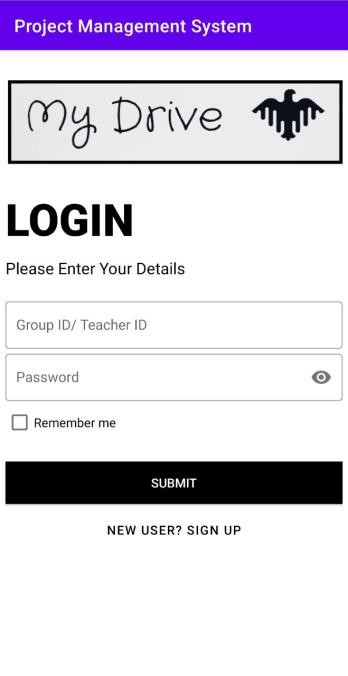
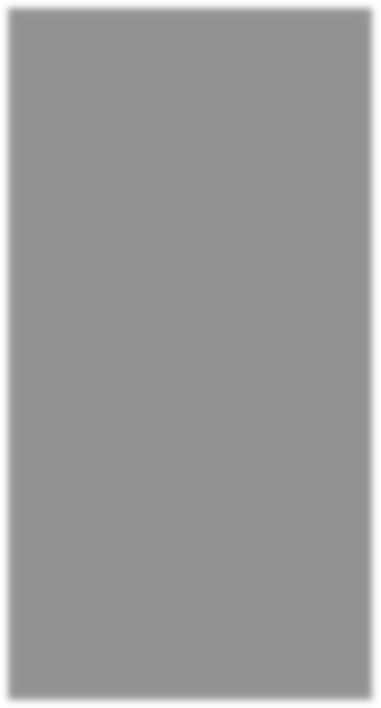
1. **Integration Testing**

Integration testing is any type of software testing that seeks to verify the interface between components against a software design. Software components may be integrated in an interactive way or altogether. A system is implementer as component cooperating with each other. Thus, component comprise of hardware and software.

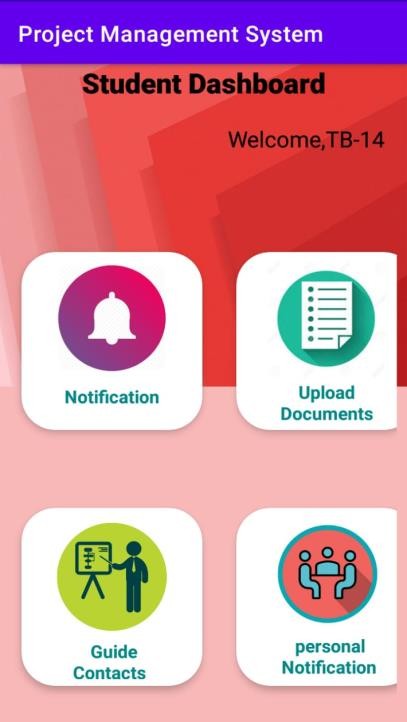
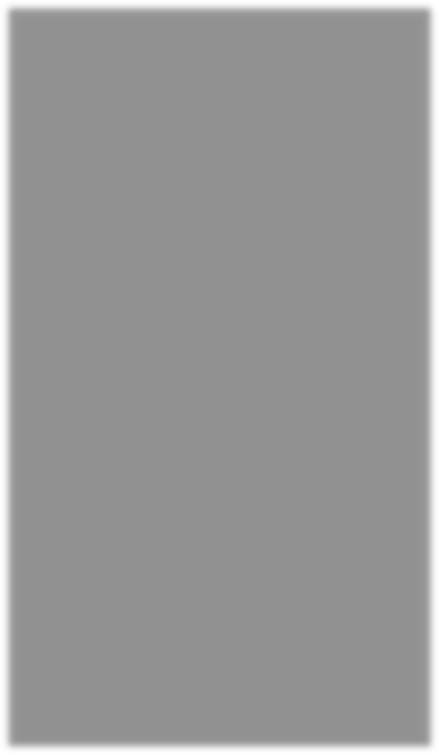
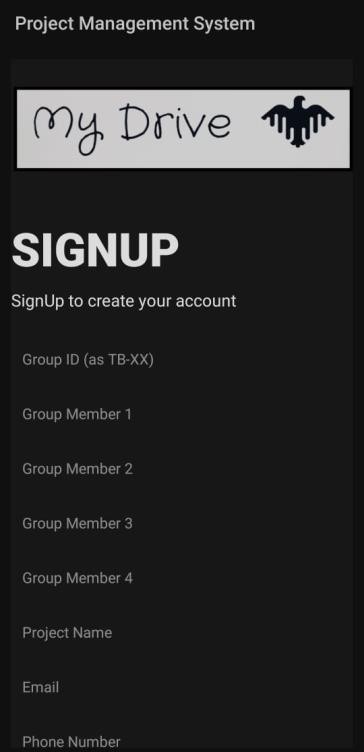
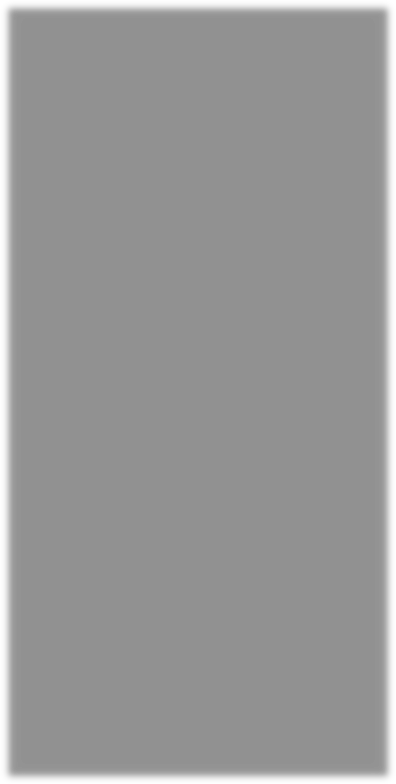
### TEST CASES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Name** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |
|  |  |  |  |  |  |
| TC01 | Registration of New Students | 1. Visit application Admin 2. Login as Admin 3. Add New   Student | Student Added Successfully | Student Added Successfully | Pass |
|  |  |  |  |  |  |
| TC02 | Login with User Credentials | 1. Visit application Admin 2. Enter Credentials 3. Click Login Button | Login Successfully | Login Successfully | Pass |
|  |  |  |  |  |  |
| TC03 | Uploading Documents | 1. Open App 2. Login & Open 3. Go to Upload document 4. Select & Submit | Submitted Successfully | Submitted Successfully | Pass |
|  |  |  |  |  |  |
| TC04 | View documents | 1. Open Android App 2. Login & Open 3. View documents | Viewed Successfully | Viewed Successfully | Pass |

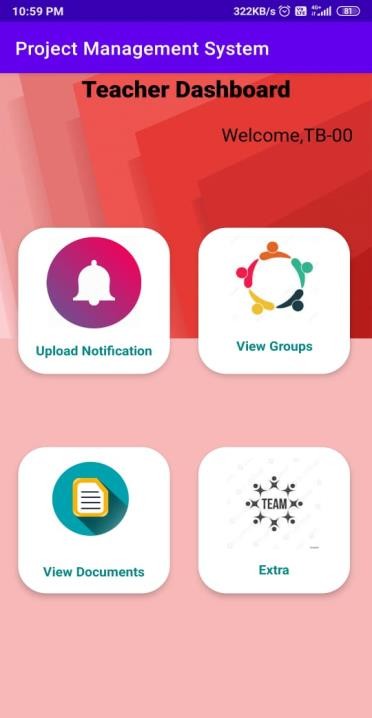
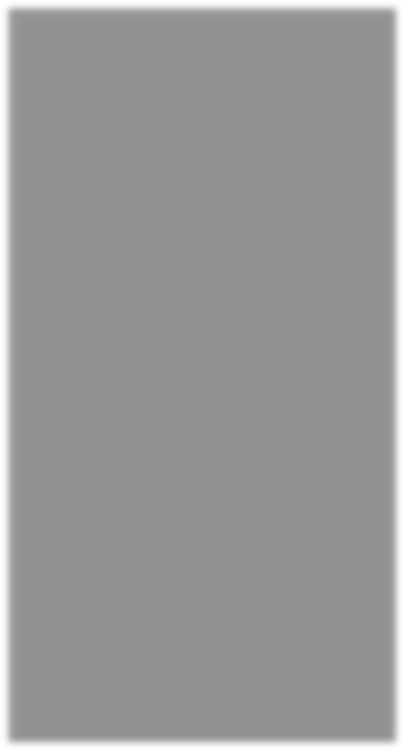
* 1. **VALIDATION OF TESTING:**
     1. **Login Activity:**



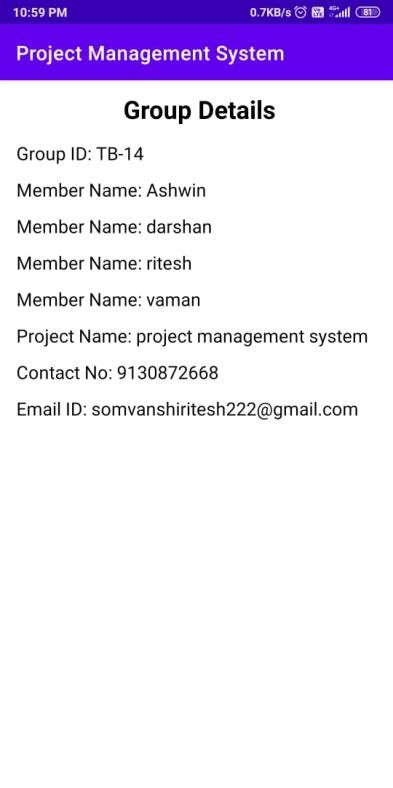
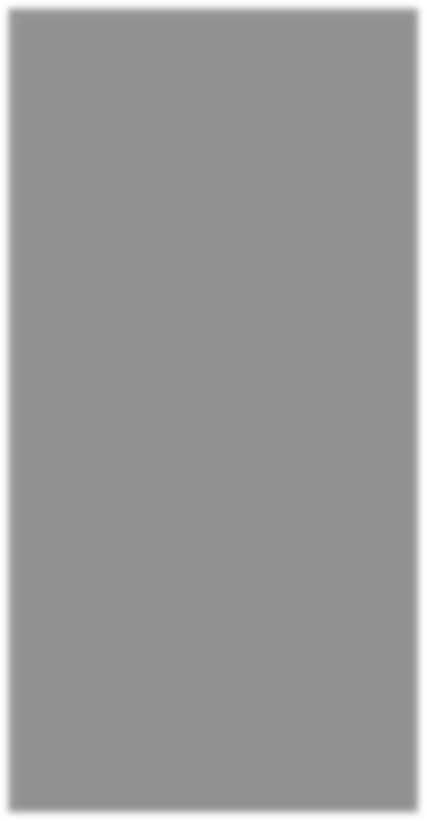
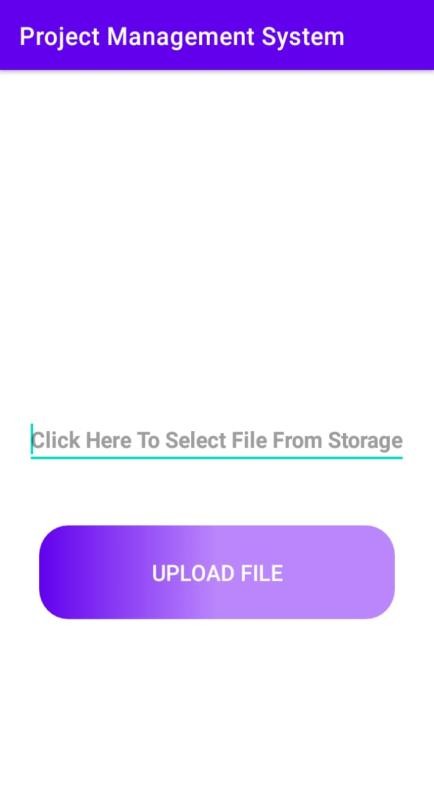
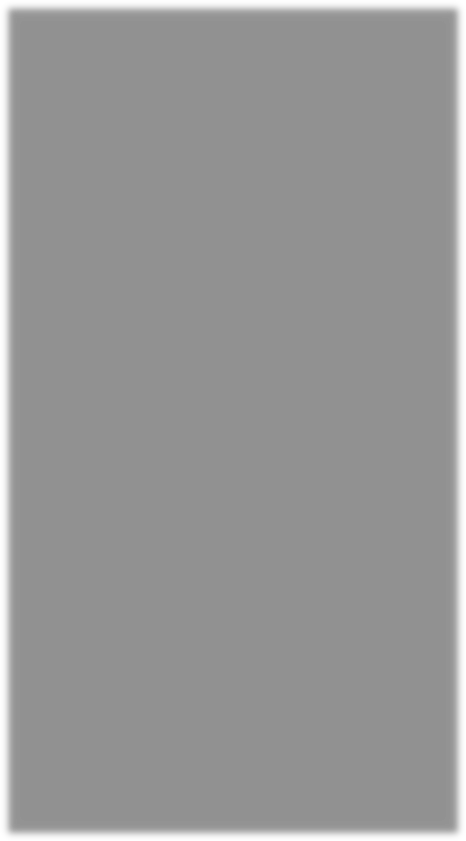
* + 1. **Registration Activity:**



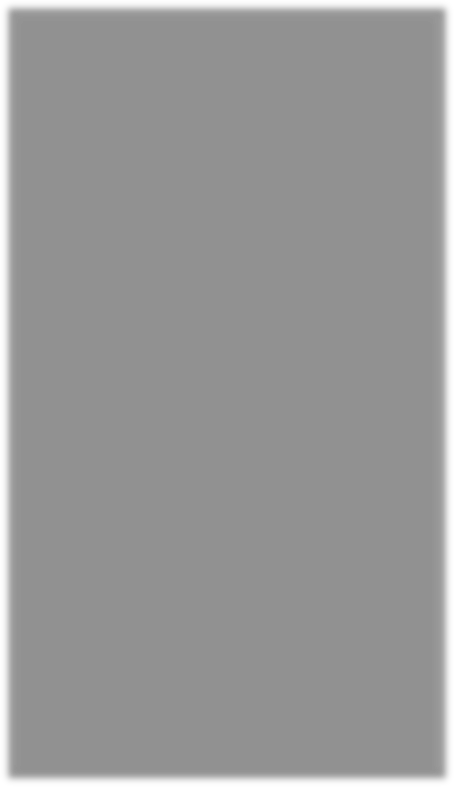
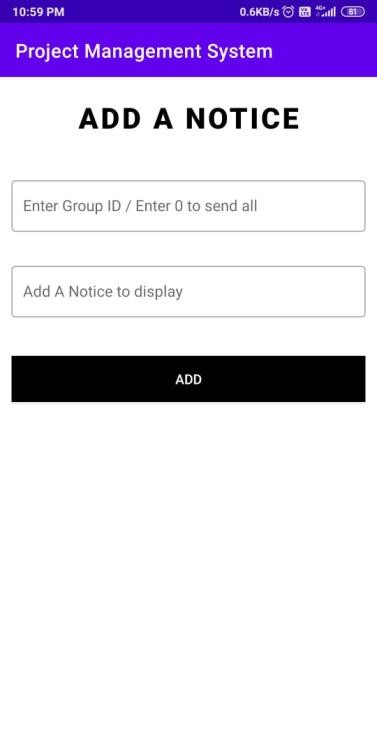
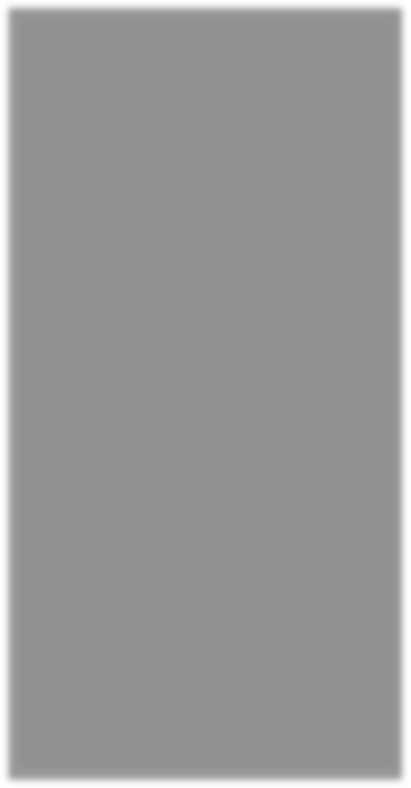
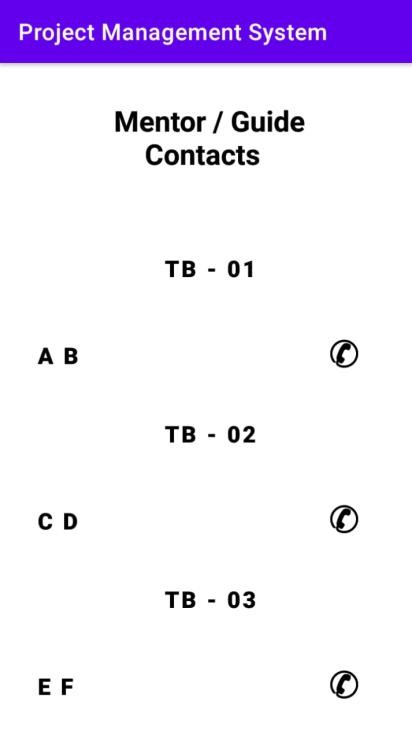
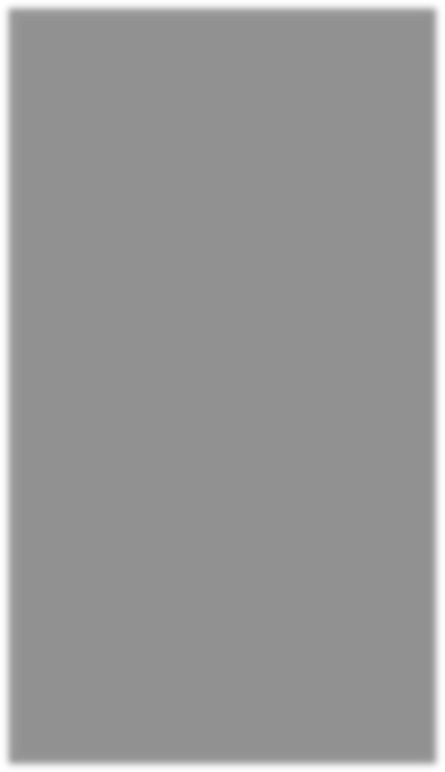
* + 1. **Dashboard: (Student view/Teacher View**)



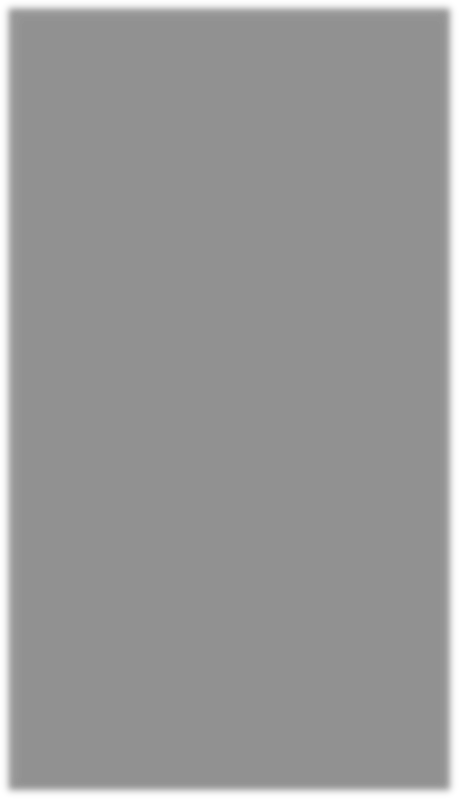
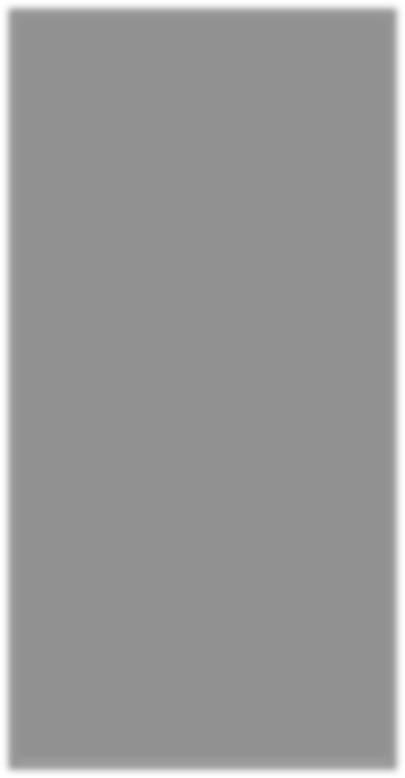
* + 1. **Main Activity Group Details And Upload Document : (Student view/Admin view)**



* + 1. **Guide Contacts & Personal Information: (Student view/Admin view)**



**5.3.6. Upload Notification &View Document (Student view/Admin view)**



**5.3.7 Notification(Student view/ Admin View)**

**1**

**Chapter 6**

**ADVANTAGES AND FUTURE SCOPE**

### Advantages :

* + - **Main motive is to reduce paper work, as every paper work on digital Platform makes easier, a smarter way to perform operations.**
    - **No meting personally, things are been done on a click of a button.**
    - **More efficient work and smart way for final project .**
    - **Save environment in the other hand by avoiding paper work.**
    - **Student and Teacher will be in secure hands.**

### Future Scope :

* + - **Schools**
    - **Collages**

**Chapter 7**

**CONCLUSION AND REFERENCES**

### CONCLUSION:

The following conclusions can be deduced from the development of the project: This system is developed to overcome the manual work.

* + - This Application is developed to overcome the manual work as much as Possible.
    - PCC can keep all the updates of students on daily basis
    - Updating of information can be easier, no complex things everything will be simple and smart
    - There will be personal attention towards student activities
    - This application Features provide best an easy interaction between student and PCC
    - System security, data security and reliability are the striking features
  1. **: Reference:**
     + Android Studio -https://[www.tutorialspoint.com/android/android\_studio.htm](http://www.tutorialspoint.com/android/android_studio.htm)
     + Java Beginner -https://[www.javatpoint.com/android-tutorial](http://www.javatpoint.com/android-tutorial)
     + Fire-base cloud -https://[www.javatpoint.com/firebase](http://www.javatpoint.com/firebase)
     + developer- https://developer.android.com/guide

Department of Computer Technology, S.K.B.P. Polytechnic, Kopargaon 44 | P a g e