A
Project Report
on

An Online Classroom System

"HybridClass"

Developed by

Rushi Raval – Department of IT, DD University
Shashwat Verma – Department of IT, DD University

Guided By

Prof. Sunil Vithlani
Department of Information Technology
D D University

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad – 387001
April - 2022

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project carried out in the subject of System Design Practice, entitled "**HybridClass**" and recorded in this report is a bonafide report of work of

1) Rushi Raval Roll No. IT135 ID No: 20ITUOD008

2) Shashwat Verma Roll No. IT149 ID No:19ITUON099

of the Department of Information Technology, semester VI. They were involved in Project work during the academic year 2021 - 2022.

Prof. Sunil Vithlani
(Project Guide)
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. (Dr.) Vipul K. Dabhi,
Head, Department of Information Technology, Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

CANDIDATE'S DECLARATION

We declare that the pre-final semester report entitled "**HybridClass - an online classroom platform**" is our own work conducted under the supervision of the guide **Prof. Sunil Vithlani.**

We further declare that to the best of our knowledge the report for B.Tech. VI semester does not contain part of the work which has been submitted either in this or any other university without proper citation.

Candidate's Signature
Rushi Raval
Student ID: 20ITUOD008

Candidate's Signature
Shashwat Verma
Student ID: 19ITUON099

ACKNOWLEDGEMENT

On the very outset of this report, we would like to extend our sincere & heartfelt obligations towards all the personages who have directly or indirectly helped in this endeavour. Without their active guidance, help, cooperation & encouragement, we wouldn't have made the headway in the project.

We are ineffably indebted to our instructor Prof. Sunil Vithlani for his guidance, encouragement and conscientiousness to accomplish this project. We are extremely thankful for his support.

We would also like to express our gratitude to our Head, Mr. Vipul Dabhi along with Dharmsinh Desai University for giving us this opportunity.

Last but not least, we would also like to acknowledge with a deep sense of reverence, our gratitude towards our family members who have always supported us morally. Any omission in this brief acknowledgement does not mean lack of gratitude.

Rushi Raval

Shashwat Verma

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

PROJECT DETAILS

Developing a Virtual Classroom system promotes a greater count of student to splurge in the field of education. It integrates the benefits of a physical classroom with the convince of a ‘no-physical-bar’ virtual learning environment, minus the commuting hazards and expenses. It will usher in the immense flexibility and sophistication in the existing learning platform structures, with the perfect blend of synchronous and asynchronous interaction. It provides a means of collaborative learning for the students.

1.1 PURPOSE

HybridClass‘s main purpose is to benefit teachers and students of universities. It enables teachers to perform various classroom functionality online. It also provides an attendance system in which teacher can generate a QR code for a lecture and student can scan the QR code in order to mark him/herself present in that lecture. Teachers can also organize quiz.

1.2 SCOPE

Student can choose courses, attend lectures, take quiz, view their attendance record, progress report, etc as per their convenience. Faculty can take lectures, upload assignments, announcement, evaluate answer sheets and can also upload lectures and other discussion in various formats as in videos, PowerPoint presentation, etc. Upload and download various assignment, college notice, student’s notices, journals, videos.

1.3 OBJECTIVE

Main motto of ours is to provide a simple and efficient way to improve access to advanced educational experience by allowing students and faculties to participate in remote learning communities using personal computers and phones and to improve the quality and effectiveness of education to support a collaborative learning.

1.4 TECHNOLOGY AND LITERATURE REVIEW

Technology:
Frontend: XML
Backend: Java
Database: Firebase

XML A markup language created as a standard way to encode data in internet-based applications. Android applications use XML to create layout files. Unlike HTML, XML is case-sensitive, requires each tag be closed, and preserves whitespace.

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need to recompile.

Firebase is a cloud-hosted database in which data is stored as JSON. The data is synchronized in real-time to every connected client. All of our clients share one Realtime Database instances and automatically receive updates with the newest data, when we build cross-platform applications with our iOS, and JavaScript SDKs.

Software Required: Android Studio.

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020.

2. PROJECT MANAGEMENT

2.1 FEASIBILITY STUDY

2.1.1 Technical Feasibility

After considering our project functionalities and technical needs we decided to go with the following technologies XML, Java, Firebase. All the mentioned technologies are open sources therefore it is feasible to use with the knowledge of basics in each of them it would be easy for us to manage the project too.

2.1.2 Time Schedule Feasibility

Initially we gathered & analyzed all the requirements by the beginning of January 2022. We prepared the SRS document and the GUI design right after that. This was followed by the required diagrams. For coding and unit testing 4 weeks and for system and integration testing another 2 weeks were covered. We are ready for its demonstration in the submission week along with the report. Being a 2 members team, we were able to complete our project in the estimated time.

2.1.3 Operational Feasibility

In the current scenario where the virtual world is the new normal, our project will excel greatly along the frequent application users who want to avail services online. We have kept the application pretty simple and handy so everyone can take advantage of it and use it without any inconvenience.

2.1.4 Implementation Feasibility

The Functional requirements of our project will be easily fulfilled by the technologies we have mentioned. All the technologies we are using are easily supported by all the browsers. Hence, the implementation is feasible.

2.2 PROJECT PLANNING

2.2.1 Project Development Approach and Justification:

The **Agile** software development methodology centers around time-boxed project cycles known as sprints. A sprint is a short period, usually two weeks, during which the team works on a set number of features called "user stories." These stories are items that the team can deliver in two weeks. As such, the sprint consists of a significantly smaller number of features than a waterfall project. Limiting the features in this manner makes for a more manageable product development and release cycle.

Figure 2.1: Agile Model- Software Development Life Cycle

2.2.2 Milestones and Deliverables

ID	Task	Timeline
1	Project Definition, Submission and approval	December
2	System Design, Layout Design	January
3	Implementation	January
4	Testing	March
5	Documentation, reporting	March

2.2.3 Roles and Responsibilities

As the project is handled by two people, the functionalities are divided as such:

Name	Analysis	Design	Coding	Testing	Documentation	Maintenance
Rushi	✔	✔	✔	✔	✔	✔
Shashwat	✔	✔	✔	✔	✔	

3. SYSTEM REQUIREMENTS STUDY

3.1 STUDY OF CURRENT SYSTEM

Application like Google Classroom, ClassDojo, Moodle are working for the good of the education. All these applications have different goals like some are e comm for study material, e comm for books and videos materials.

3.2 USER CHARACTERISTICS

1. Teacher – Perform various classroom functionalities as teacher role.
2. Student – Registers himself at the application and avail the features of the application.

3.3 HARDWARE AND SOFTWARE REQUIREMENTS

3.3.1 Software requirements

For all the actors the basic software requirements are as follows

- Good internet connection.

3.3.2 Hardware requirements

For all the actors the basic hardware requirements are as follows

- Device to run the application

3.4 CONSTRAINTS

3.4.1 Hardware Constraints

- As it is a application, internet is required for accessing different
- Response time can vary depending on the net connection.

3.4.2 Higher Language Order Constraints

The application requires the front end to be the Android.

3.4.3 Reliability Requirements

The application must adhere to the reliability requirements as needed and should run smoothly on devices.

3.4.4 Safety and Security Consideration

The application does not access any feature of the mobile which may cause it to compromise on the user's security or personal information. No data of user except location is used by the application; hence, data theft is prevented.

3.4.5 Criticality of the Application

The application does little work if there is no working network or data connection available.

4. SYSTEM ANALYSIS

4.2.1 REQUIREMENTS OF NEW SYSTEM (SRS)

(Teacher or Student)

Sign Up (Click on "Register" option in menu bar)

Description: For the users to access some features of this application registration is a must.

Pre-Condition: The user should not be already registered. It has to be a new account**.**

Input: The user enter detail: Username, First Name, Last Name, Email and Password

Output: Redirected to home page.

Post-Condition: The data is stored in the database.

Sign In (Log In)

Description: Only registered users can have access.

Pre-Condition: The user should be already registered.

Input: The user enters username and password.

Processing: Checks whether the user is already registered and password is correct or not.

Output: User logged in.

Post-Condition: Redirected to home page.

Create Classroom

Description: Teacher can create new classroom for different subjects for students to join.

Pre-Condition: That class shouldn't already exist

Input: Clicks on "Create" button

Output: Creates a new classroom and a unique joining code for students

Post-Condition: User can continue with any activity on application

View/Give Attendance

Description: Teacher can View/Give attendance to Students for that particular subject.

Input: Click on "View/Give Attendance" to get detailed information of student attendance

Output: Page with details of the attendance of students for different subjects

Post-Condition: User can view and edit attendance of students

Create/Edit E-Notice Board

Description: Create or Edit notice which can be a common notice or for a particular subject.

Input: Click the "Create Notice" button.

Output: Acknowledgement.

Post-Condition: Teacher and students can view the E-notice. Teacher can edit E-notice

Give Study Material

Description: Teachers can provide/upload study material for different subjects.
Input: Click on "Upload" button
Output: Redirect to main page.
Post-Condition: User can continue with any activity on application.

Organize Quiz

Description: Teacher can create a quiz for subjects for the students to attend
Input: Click on "Organize Quiz" button and fill out required question and options.
Output: Redirects to quiz tab.
Post-Condition: Quiz created and Teacher can assign it to students.

Give Marks/Performance

Description: Teacher can give marks for the quiz attended by the students.
Pre-Condition: Quiz should already be attended by the students.
Input: Click on "Give Marks"
Output: Redirect to quiz tab.
Post-Condition: Based on the input result will be displayed.

Give Performance Review

Description: Teacher can give individual performance review for each subject to each student.
Pre-Condition: Students should have taken the quiz created by the teacher.
Input: Click on "Give Review" button.
Output: Review of each student would be uploaded
Post-Condition: Based on the input result will be displayed.

Log Out

Description: To leave the application, logout/checkout from the application.
Pre-Condition: Users should be logged in.
Input: Click the 'Logout' button.
Output: User is logged out and redirected to the login page.
Post-Condition: None.

(For Student Only)

Join Classroom

Description: To join a classroom, student has to input the joining code which the teacher would have shared with them.
Pre-Condition: Student must not be in that classroom already
Input: Click the "Join" button.
Output: Joined successfully upon entering the correct code
Post-Condition: Student can now access the materials/notices/quiz of that classroom, uploaded by the teacher.

Mark Attendance

Description: Students can mark their attendance for each individual subjects by scanning the QR code created for attendance with their phone.
Input: Click "Mark Attendance" button
Output: Attendance for the subject will be marked and updated in the database
Post-Condition: Redirected to classroom page

View Attendance

Description: Student can view their attendance.
Input: Click "View Attendance" button
Output: Student can view their attendance for respective subjects
Post-Condition: User can continue with any activity on application

View E-Notice Board

Description: View notice which can be a common notice or for a particular subject.
Input: Click the "View Notice" button.
Output: Notice is displayed
Post-Condition: Students can view the E-notice uploaded by teacher.

View Study Material

Description: Student can view the study material of different subject provided by the teacher.
Input: Click on "View Material" button
Output: Study material/notes are displayed for a particular subject
Post-Condition: User can continue with any activity on application

Attend Quiz

Description: Student can attend a quiz of a subjects created by the teacher.
Input: Click on "Attend Quiz" button
Output: Redirects to quiz tab.
Post-Condition: Student can attend the quiz.

View Marks/Performance

Description: Student can view marks for the quiz attended by them
Pre-Condition: Quiz should already be attended by the students.
Input: Click on "View Marks" button
Output: Marks are displayed
Post-Condition: Based on the input result will be displayed

View Performance Review

Description: Students can view individual performance review for each subjects.
Pre-Condition: Students must have taken the quiz.
Input: Click on "View Review" button
Output: Review of students will be displayed
Post-Condition: Based on the input result will be displayed.

Log Out

Description: To leave the application, logout/checkout from the application.
Pre-Condition: Users should be logged in.
Input: Click the 'Logout' button.
Output: User is logged out and redirected to the login page.
Post-Condition: None.

4.2.1 Use-case Diagram

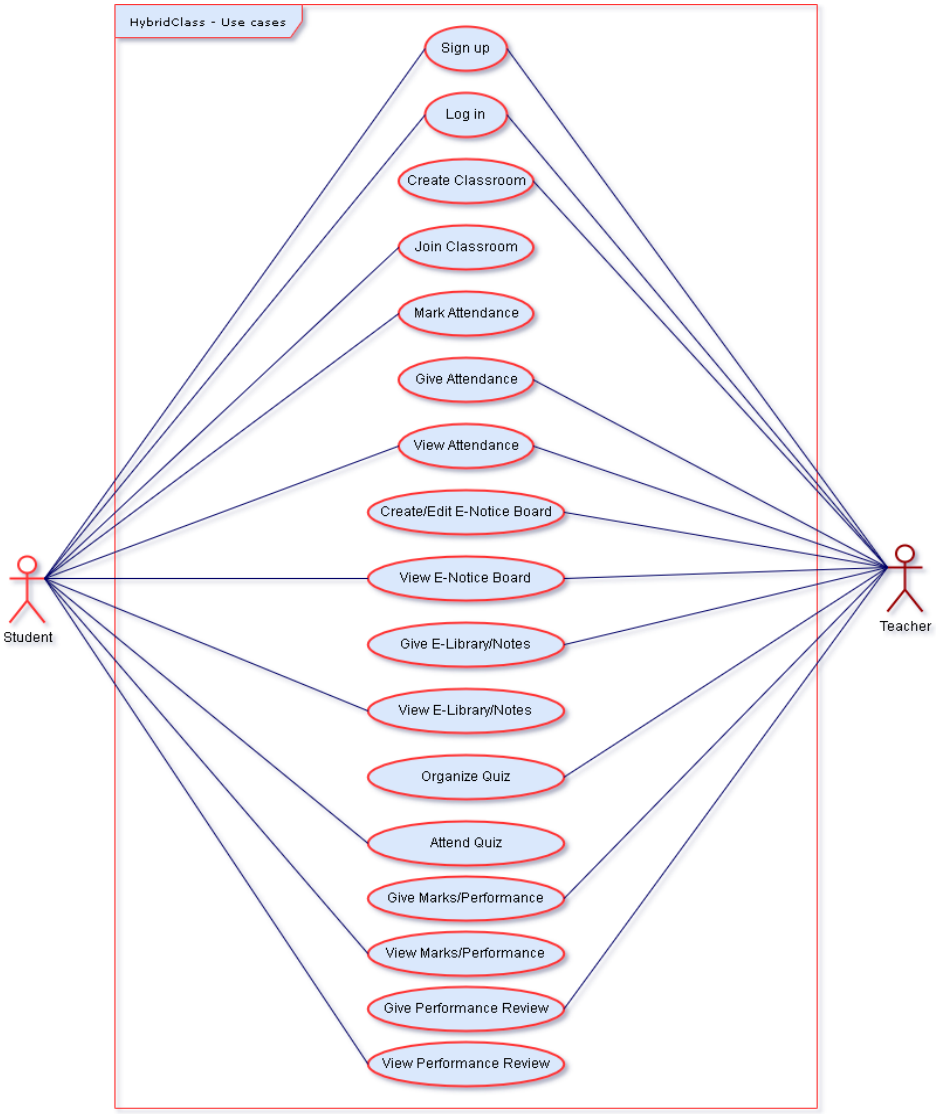


Figure 4.2.1: Use-Case Diagram

5. SYSTEM DESIGN

5.1 SEQUENCE DIAGRAM

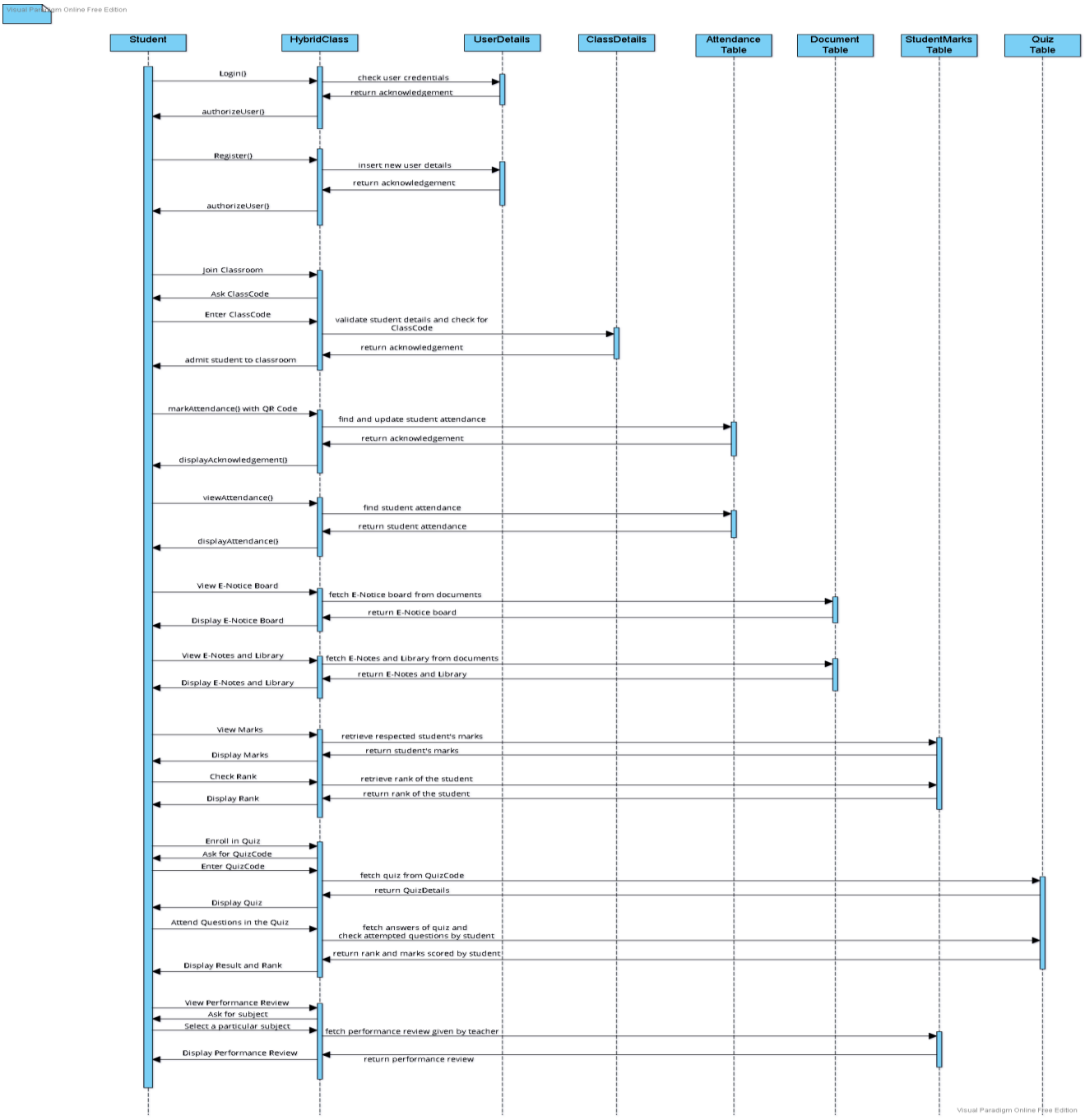


Figure 5.1.1: Sequence Diagram for Register Student

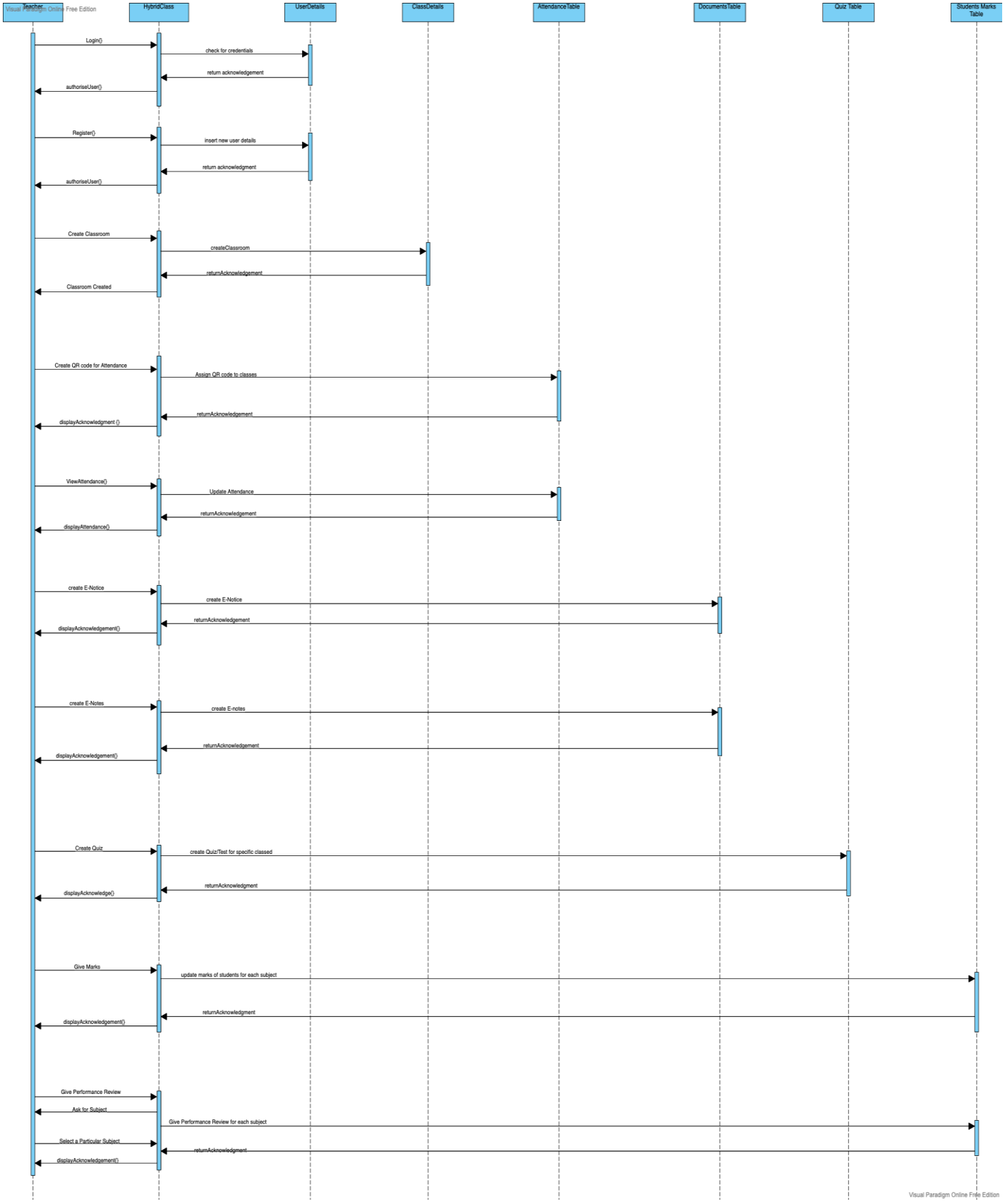


Figure 5.1.2: Sequence Diagram of Teacher

5.2 ACTIVITY DIAGRAM

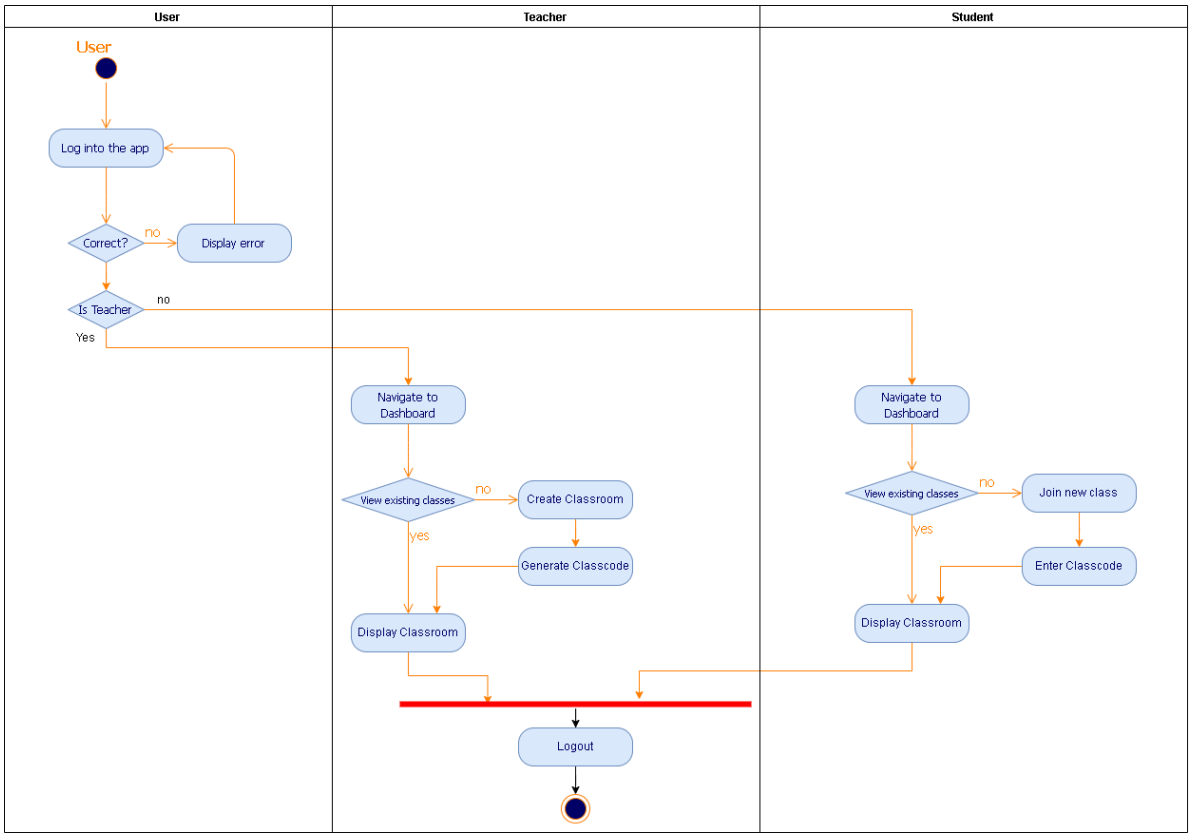


Figure 5.2.1: Activity Diagram of Classroom Functionality

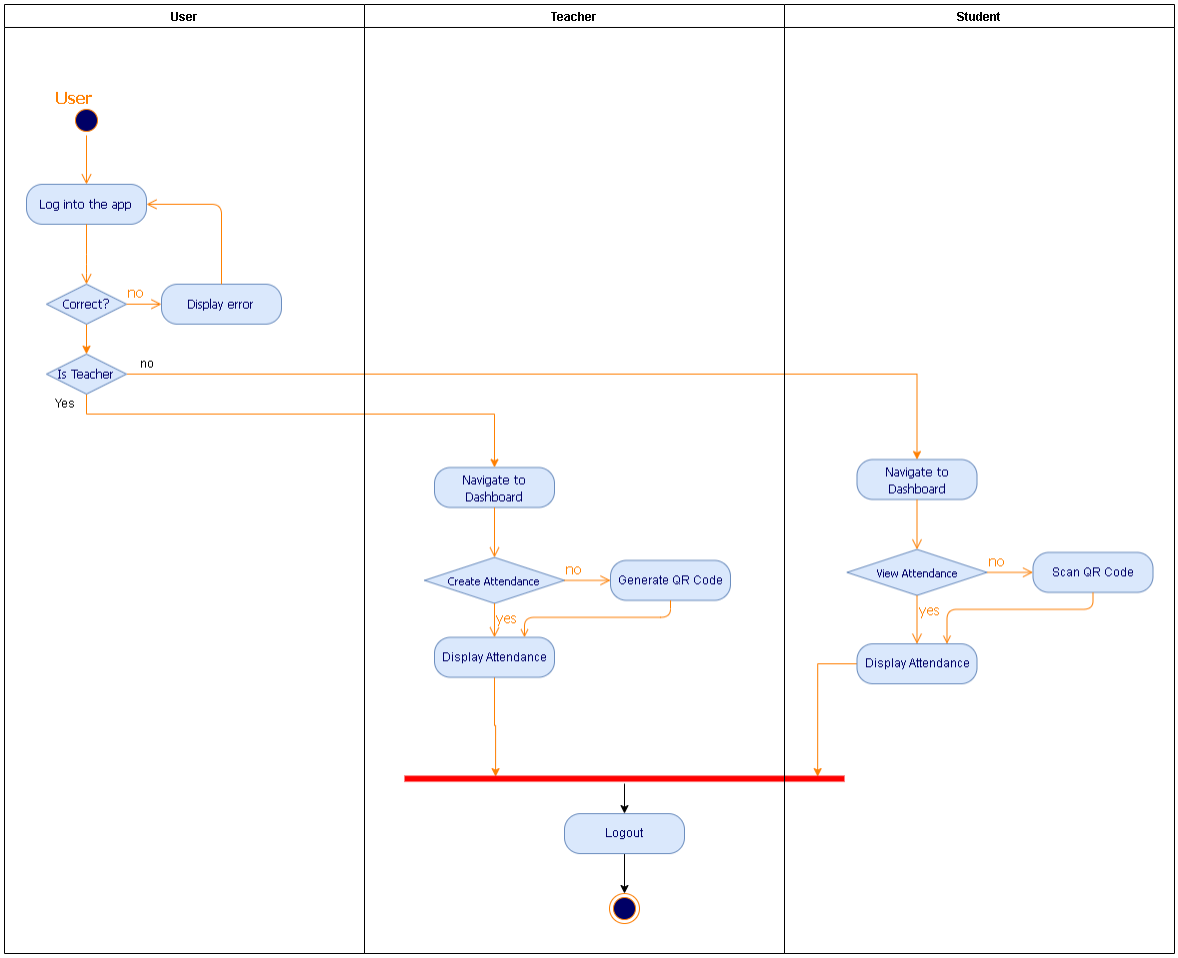


Figure 5.2.2: Activity Diagram of Attendance Functionality

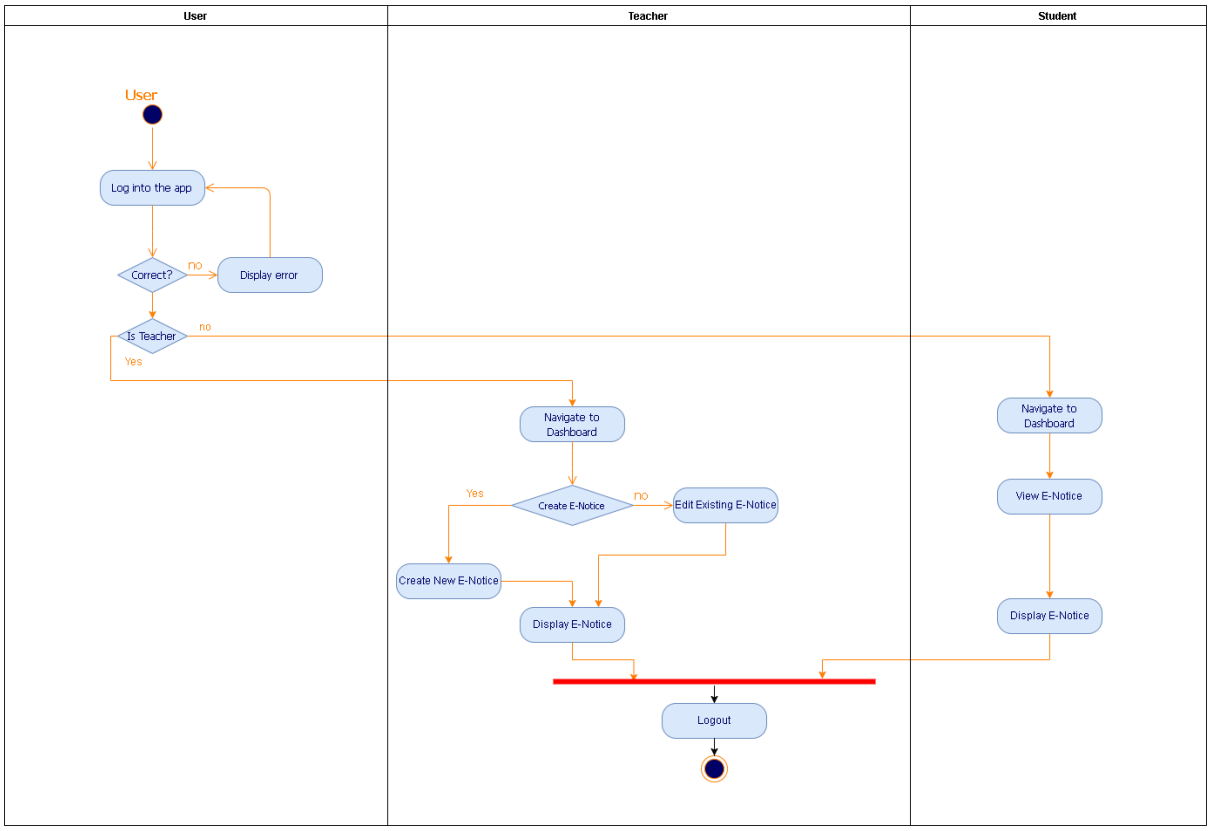


Figure 5.2.3: Activity Diagram of Notice Functionality

5.3 CLASS DIAGRAM

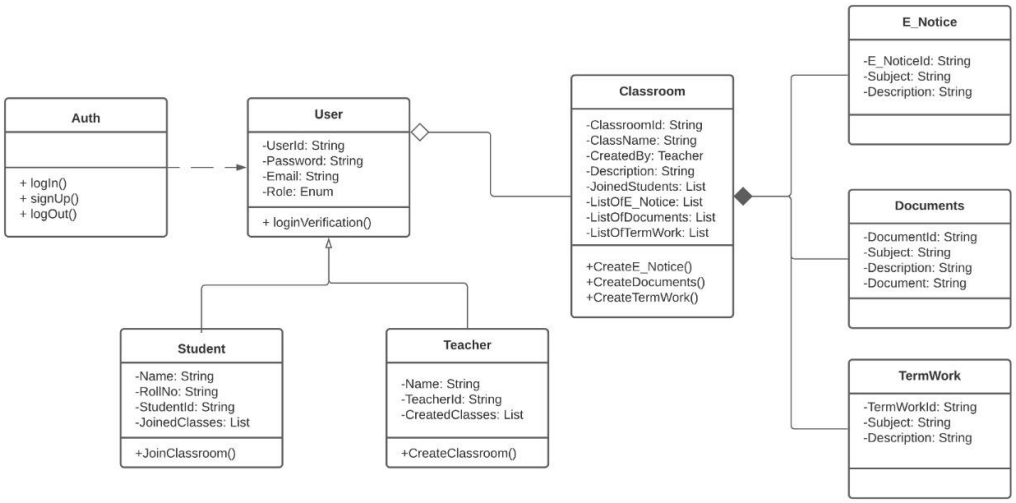


Figure 5.3.1: Class Diagram

5.4 DFD DIAGRAM

Level 0:

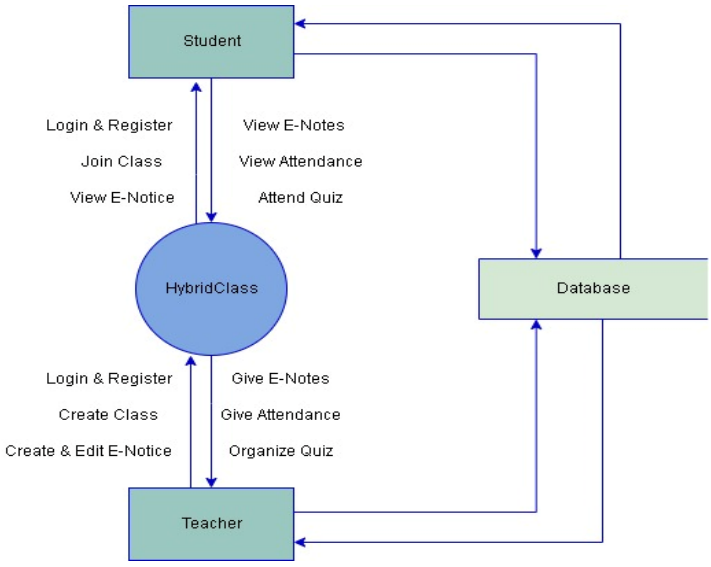


Figure 5.4.1: DFD Level 0

Level 1:

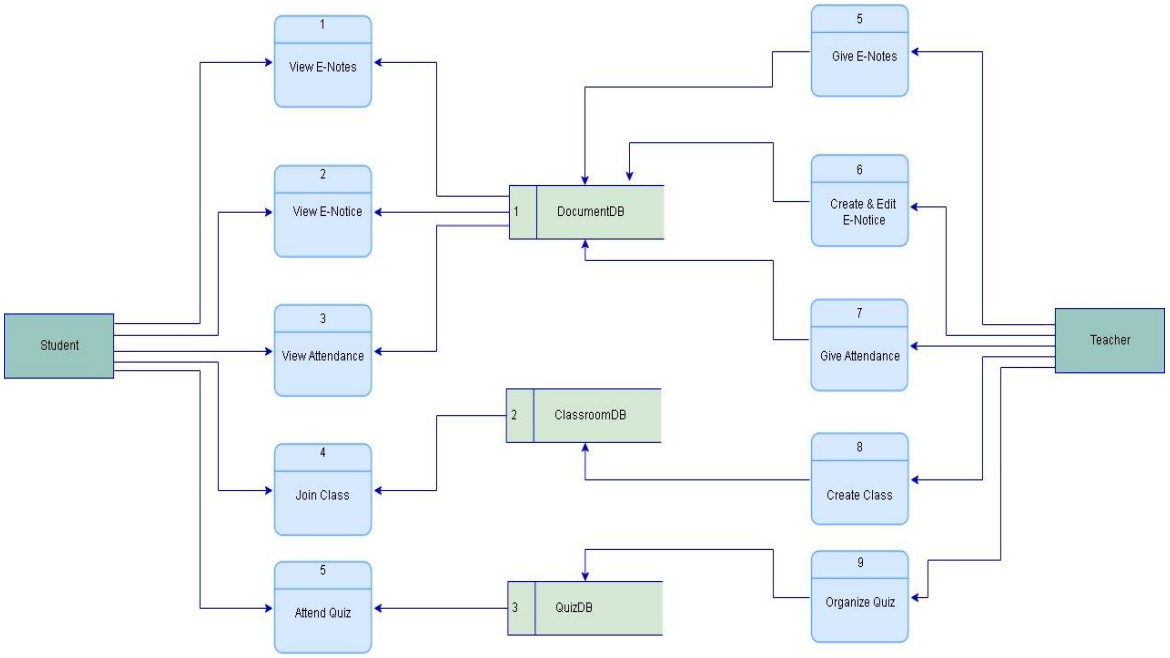


Figure 5.4.2: DFD Level 1

6. IMPLEMENTATION PLANNING

6.1 IMPLEMENTATION ENVIRONMENT

During the complete implementation we have worked on Android studio. Android Studio is a free source-code GUI (Graphical User Interface) editor for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. It also provides extension to work in a team with a multi user experience. Even compiling and deploying of the application was done easily with the help of Android studio.

For the database implementation, We have used Firebase. Firebase is a powerful, open source object-relational database system Specially designed for Android Related Web Projects.

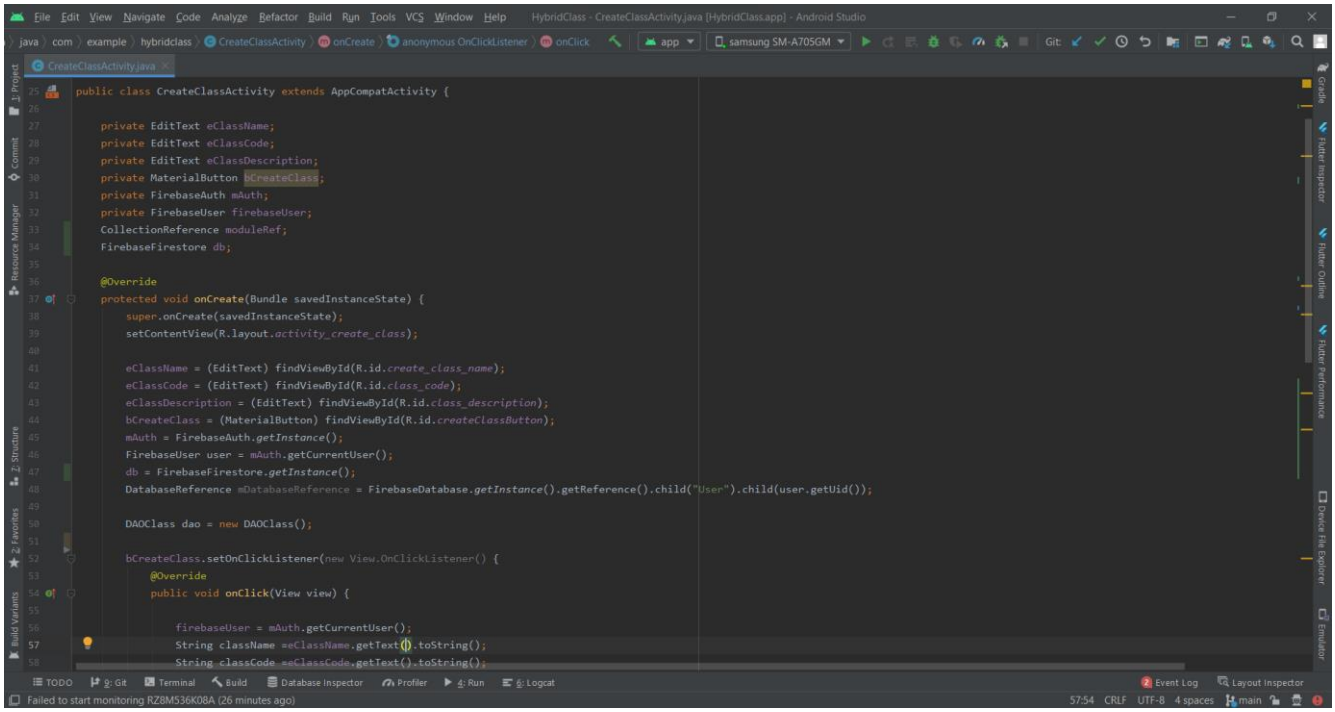
6.2 PROGRAMS/MODULES SPECIFICATION

- Visual Studio Code
- Firebase database Server
- OS: Windows

6.3 CODING STANDARDS

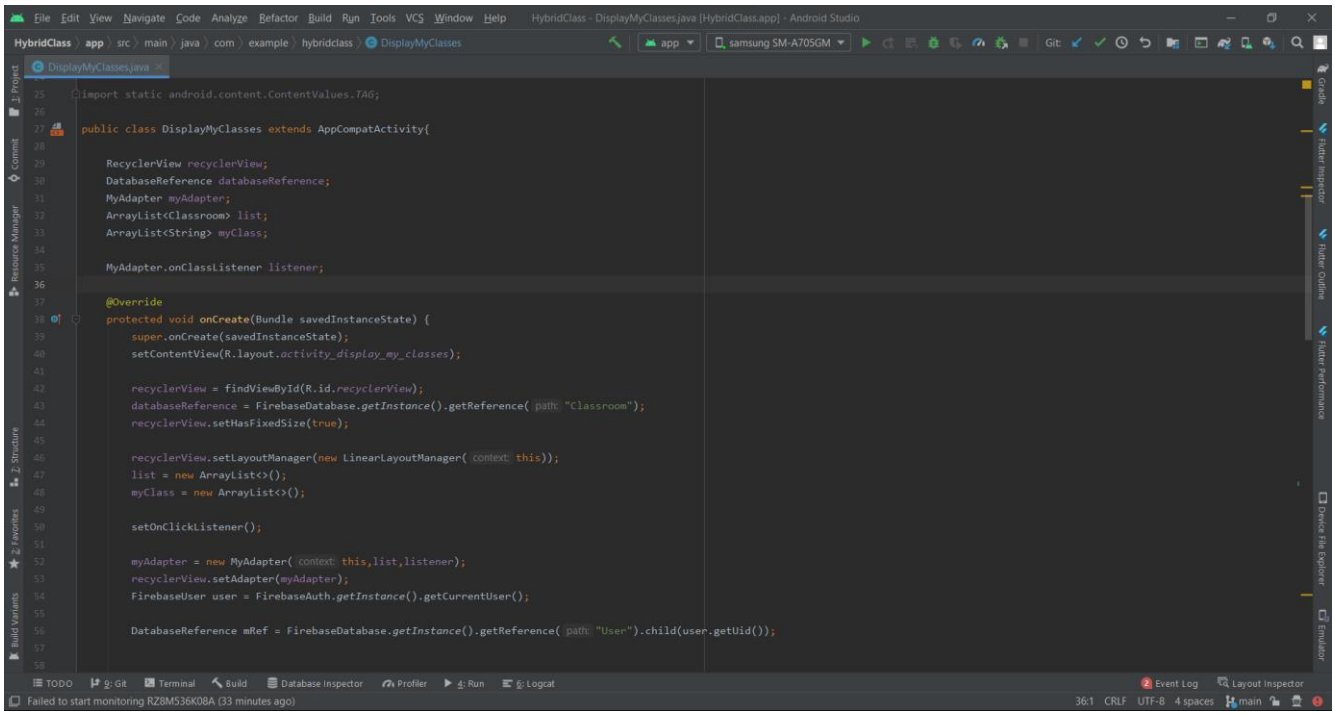
To make the system coding easy, easy to remember and reducing the chances of errors ES Lint is used as extension at the time of coding of the application which is worldwide accepted coding standard.

6.4 CODING Snippets



```
public class CreateClassActivity extends AppCompatActivity {

    private EditText eClassName;
    private EditText eClassCode;
    private EditText eClassDescription;
    private MaterialButton bCreateClass;
    private FirebaseAuth mAuth;
    private FirebaseUser firebaseUser;
    CollectionReference moduleRef;
    FirebaseFirestore db;

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

        eClassName = (EditText) findViewById(R.id.create_class_name);
        eClassCode = (EditText) findViewById(R.id.class_code);
        eClassDescription = (EditText) findViewById(R.id.class_description);
        bCreateClass = (MaterialButton) findViewById(R.id.createClassButton);
        mAuth = FirebaseAuth.getInstance();
        FirebaseUser user = mAuth.getCurrentUser();
        db = FirebaseFirestore.getInstance();
        DatabaseReference mDatabaseReference = FirebaseDatabase.getInstance().getReference().child("User").child(user.getUid());

        DAOClass dao = new DAOClass();

        bCreateClass.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                firebaseUser = mAuth.getCurrentUser();
                String className =eClassName.getText().toString();
                String classCode =eClassCode.getText().toString();
```

Figure 6.4.1: Create Class Functionality



```
import static android.content.ContentValues.TAG;

public class DisplayMyClasses extends AppCompatActivity{

    RecyclerView recyclerView;
    DatabaseReference databaseReference;
    MyAdapter myAdapter;
    ArrayList<Classroom> list;
    ArrayList<String> myClass;

    MyAdapter.onClassListener listener;

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

        recyclerView = findViewById(R.id.recyclerView);
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "Classroom");
        recyclerView.setHasFixedSize(true);

        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
        list = new ArrayList<>();
        myClass = new ArrayList<>();

        setOnClickListener();

        myAdapter = new MyAdapter( context: this,list,listener);
        recyclerView.setAdapter(myAdapter);
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

        DatabaseReference mRef = FirebaseDatabase.getInstance().getReference( path: "User").child(user.getUid());
```

Figure 6.4.2: Display Class Functionality

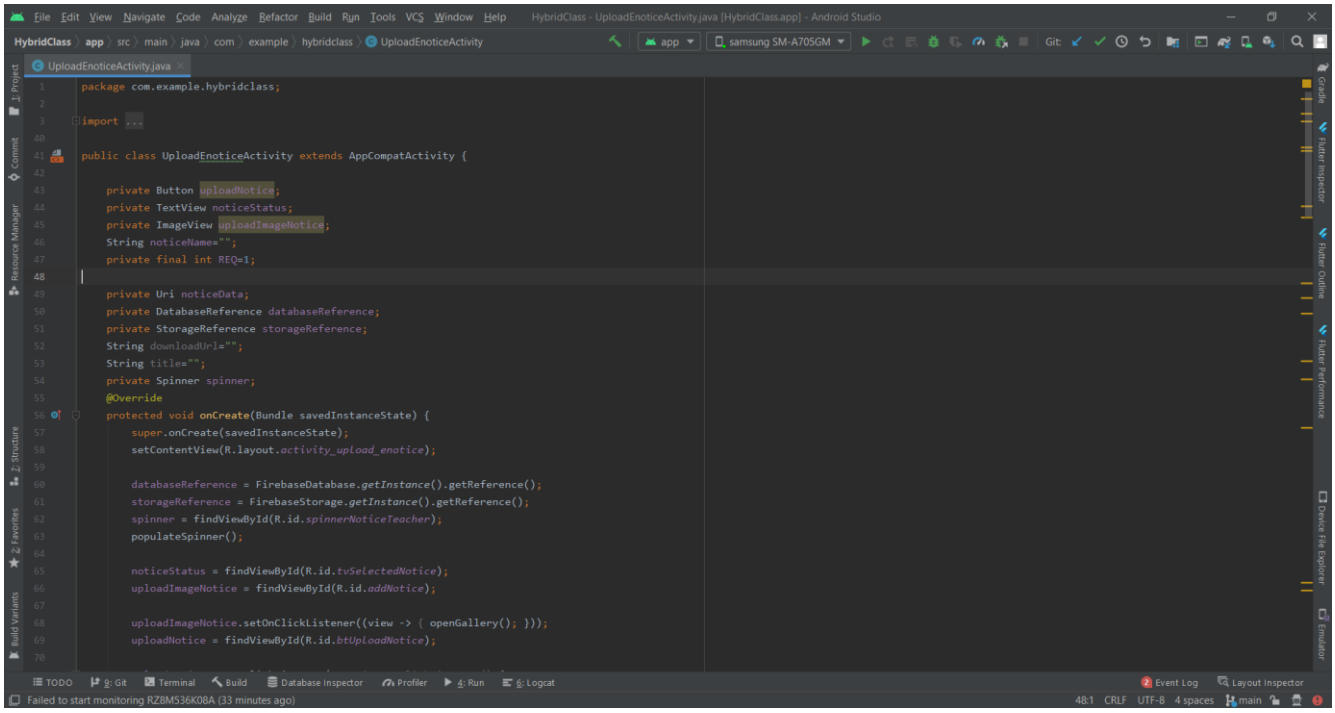


Figure 6.4.3: Upload E- Notice Functionality

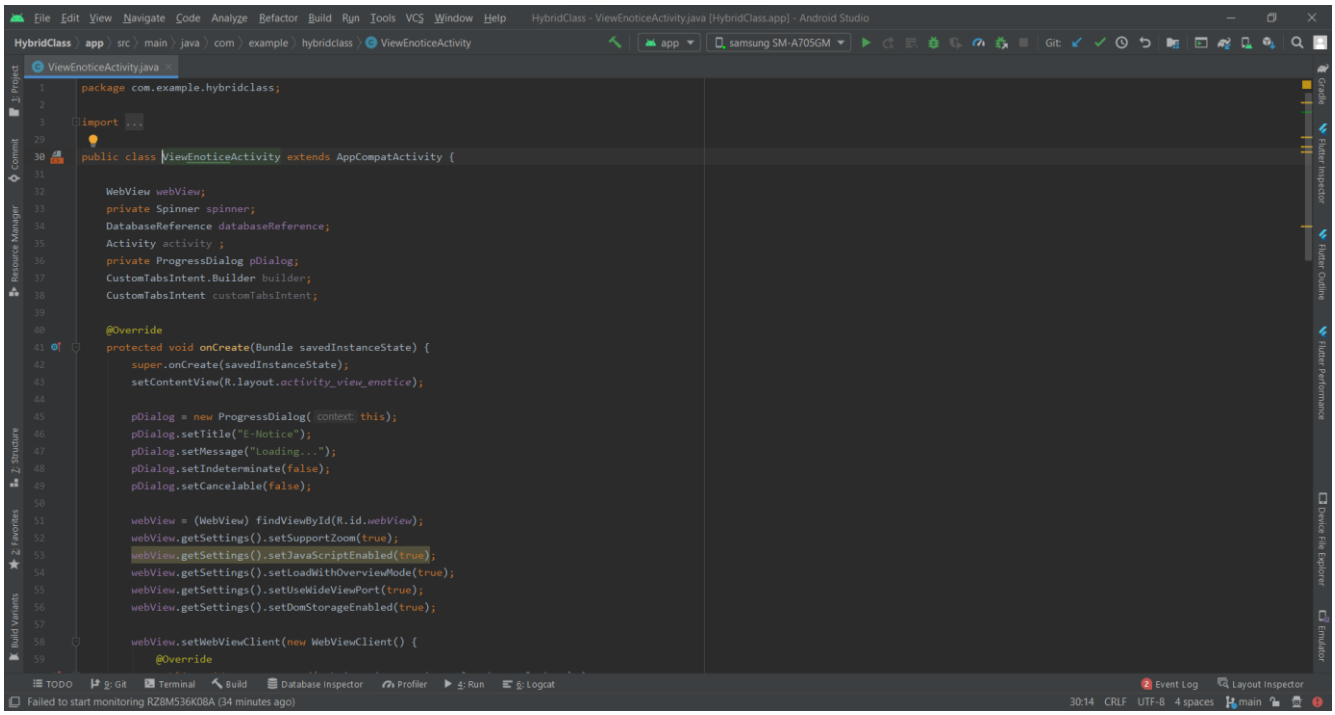
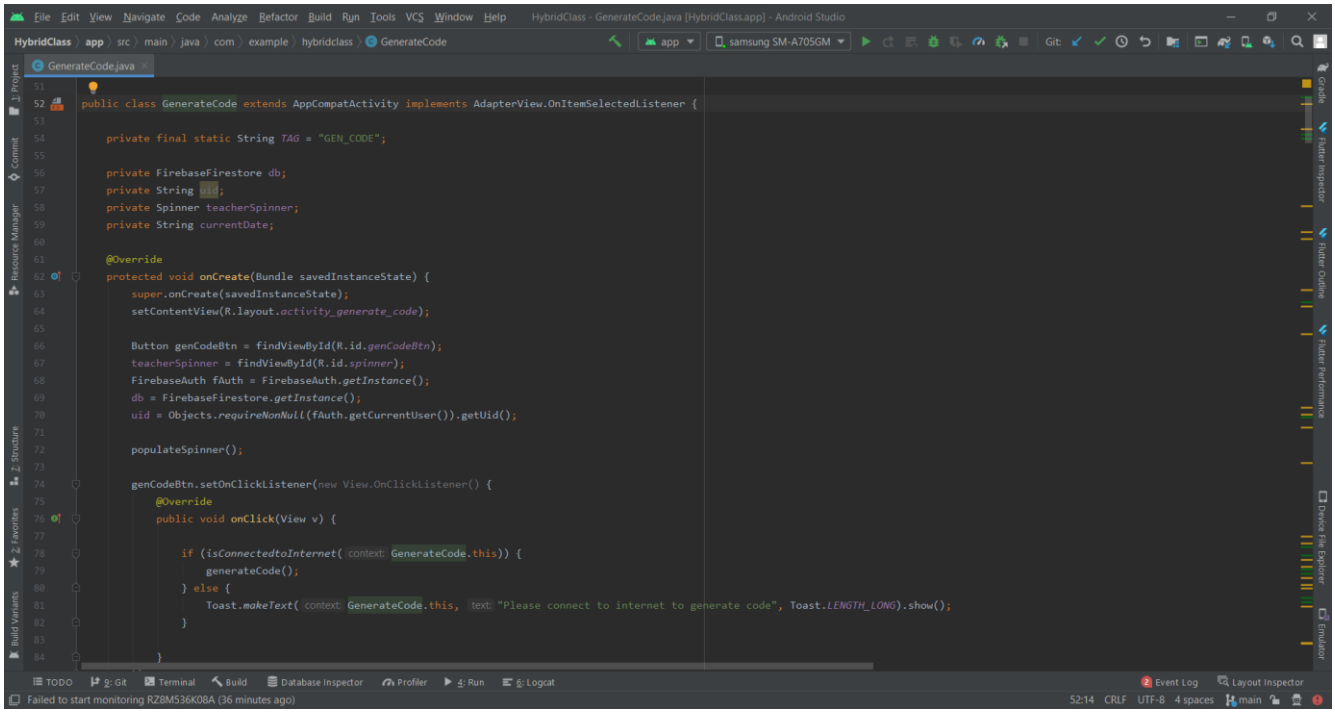
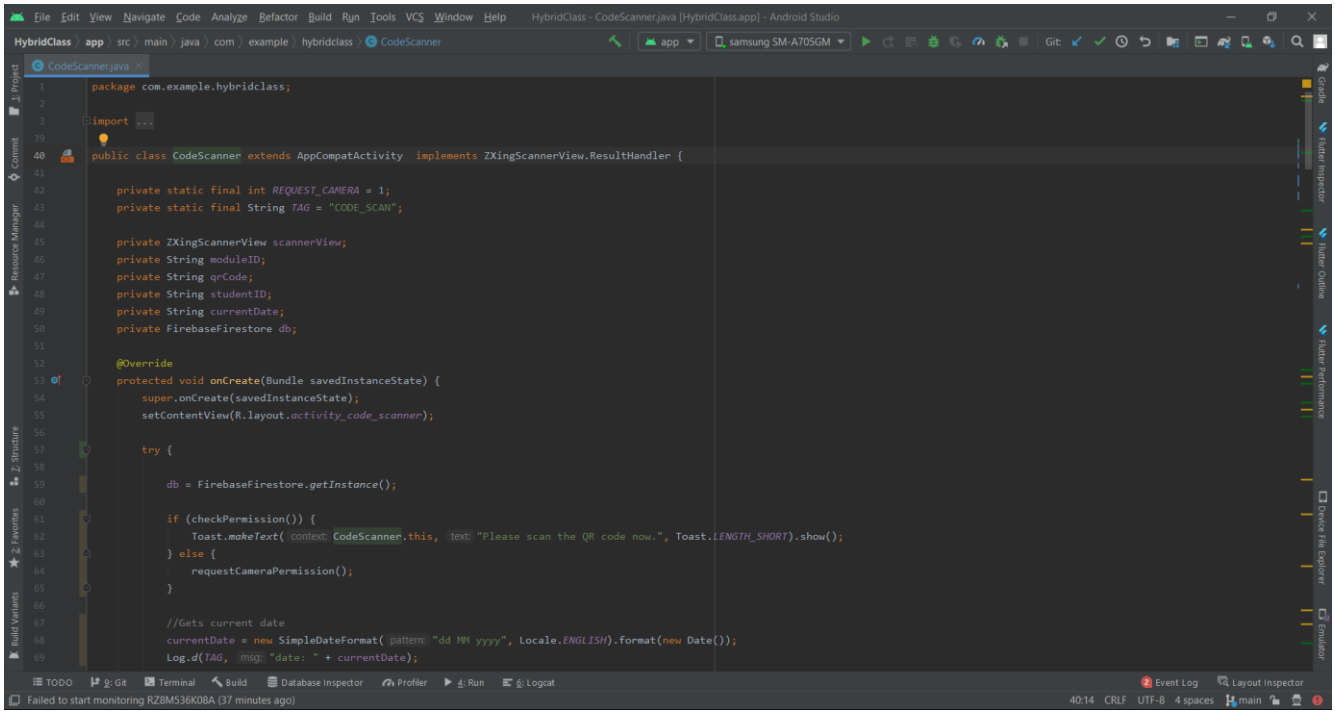


Figure 6.4.4: View E-Notice Functionality



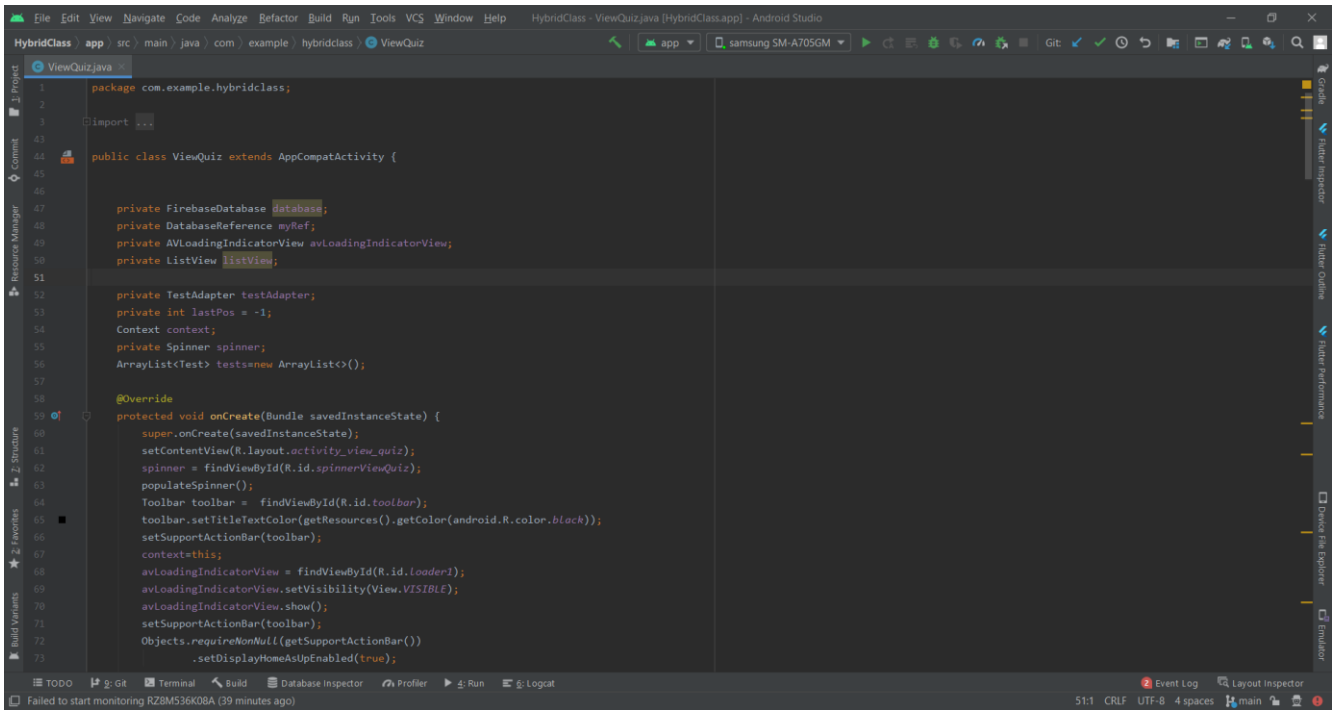
```
public class GenerateCode extends AppCompatActivity implements AdapterView.OnItemSelectedListener {

    private final static String TAG = "GEN_CODE";

    private FirebaseFirestore db;
    private String uid;
    private Spinner teacherSpinner;
    private String currentDate;

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

        Button genCodeBtn = findViewById(R.id.genCodeBtn);
        teacherSpinner = findViewById(R.id.spinner);
        FirebaseAuth fAuth = FirebaseAuth.getInstance();
        db = FirebaseFirestore.getInstance();
        uid = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();

        populateSpinner();

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

                if (isConnectedtoInternet( context: GenerateCode.this)) {
                    generateCode();
                } else {
                    Toast.makeText( context: GenerateCode.this, text: "Please connect to internet to generate code", Toast.LENGTH_LONG).show();
                }

            }
```

Figure 6.4.5: Generate QR Code Functionality



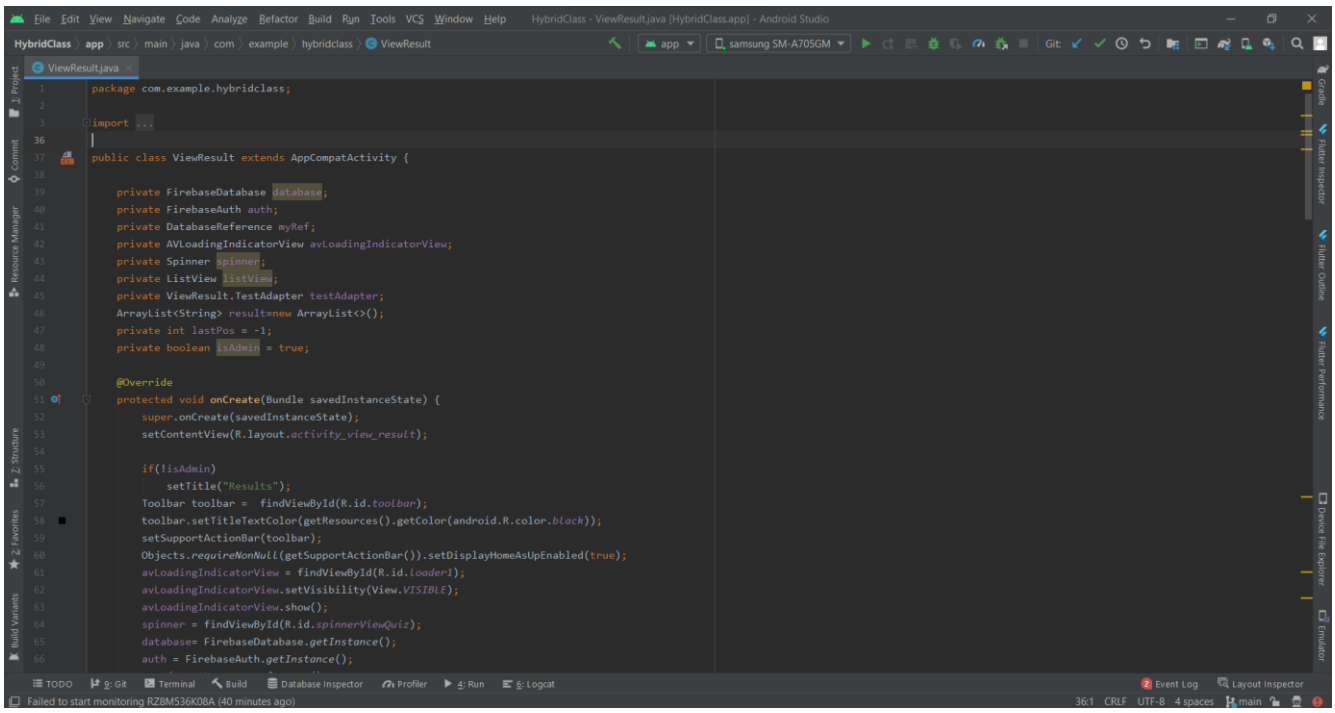
```
package com.example.hybridclass;

import ...

public class CodeScanner extends AppCompatActivity  implements ZXingScannerView.ResultHandler {

    private static final int REQUEST_CAMERA = 1;
    private static final String TAG = "CODE_SCAN";

    private ZXingScannerView scannerView;
    private String moduleID;
    private String qrCode;
    private String studentID;
    private String currentDate;
    private FirebaseFirestore db;

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

        try {

            db = FirebaseFirestore.getInstance();

            if (checkPermission()) {
                Toast.makeText( context: CodeScanner.this, text: "Please scan the QR code now.", Toast.LENGTH_SHORT).show();
            } else {
                requestCameraPermission();
            }

            //Gets current date
            currentDate = new SimpleDateFormat( pattern: "dd MM yyyy", Locale.ENGLISH).format(new Date());
            Log.d(TAG,  msg: "date: " + currentDate);
```

Figure 6.4.6: Scan QR Code Functionality



```
package com.example.hybridclass;

import ...

public class ViewQuiz extends AppCompatActivity {


    private FirebaseDatabase database;
    private DatabaseReference myRef;
    private AVLoadingIndicatorView avLoadingIndicatorView;
    private ListView listView;

    private TestAdapter testAdapter;
    private int lastPos = -1;
    Context context;
    private Spinner spinner;
    ArrayList<Test> tests=new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_quiz);
        spinner = findViewById(R.id.spinnerViewQuiz);
        populateSpinner();
        Toolbar toolbar =  findViewById(R.id.toolbar);
        toolbar.setTitleTextColor(getResources().getColor(android.R.color.black));
        setSupportActionBar(toolbar);
        context=this;
        avLoadingIndicatorView = findViewById(R.id.loader1);
        avLoadingIndicatorView.setVisibility(View.VISIBLE);
        avLoadingIndicatorView.show();
        setSupportActionBar(toolbar);
        Objects.requireNonNull(getSupportActionBar())
                .setDisplayHomeAsUpEnabled(true);
```

Figure 6.4.7: View Quiz Functionality



```
package com.example.hybridclass;

import ...

public class ViewResult extends AppCompatActivity {

    private FirebaseDatabase database;
    private FirebaseAuth auth;
    private DatabaseReference myRef;
    private AVLoadingIndicatorView avLoadingIndicatorView;
    private Spinner spinner;
    private ListView listView;
    private ViewResult.TestAdapter testAdapter;
    ArrayList<String> result=new ArrayList<>();
    private int lastPos = -1;
    private boolean isAdmin = true;

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

        if(!isAdmin)
            setTitle("Results");
        Toolbar toolbar =  findViewById(R.id.toolbar);
        toolbar.setTitleTextColor(getResources().getColor(android.R.color.black));
        setSupportActionBar(toolbar);
        Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);
        avLoadingIndicatorView = findViewById(R.id.loader1);
        avLoadingIndicatorView.setVisibility(View.VISIBLE);
        avLoadingIndicatorView.show();
        spinner = findViewById(R.id.spinnerViewQuiz);
        database= FirebaseDatabase.getInstance();
        auth = FirebaseAuth.getInstance();
```

Figure 6.4.8: View Result Functionality

7. TESTING

7.1 TESTING PLAN

Software Testing

Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they don't happen when they should. It is oriented to 'detection'.

The Need For Testing:

No matter how good a programmer is, no application will ever be one hundred percent correct. Testing was important to us in order to ensure that the application works as efficient as possible and conforms to the needs of the system. Testing was carried out throughout the development of the application, not just the application has been developed, as at this stage it took a great deal of effort to fix any bugs or design problems that were occurred.

7.2 TESTING STRATEGY

When our application was configured and customized in the system, the test was observed that this configuration or customization does not cause any improper processing or violation. The following care was taken when the application was developed at the local machine. The interface may have something not proper, which can be tested by this checklist:

- Number of input parameter equal to number of arguments
- Parameter and argument attributes match
- Number of arguments transmitted to called forms equal to number of parameters
- Attributes of arguments transmitted to called forms to attributes of parameters
- Number of attributes and order of arguments to built-in functions correct
- The local data structures for a form are common source of errors. The following types of errors should be searched for,
- Improper or inconsistent typing
- Erroneous initialization or default values
- Incorrect places name
- Underflow, overflow and addressing exception

- As far as unit testing is concerned, we did it at the time of coding in an informal but extensive way, so as to reduce number of problems arising out of incorrect syntax, incorrect variable, function names etc.
- Close the database connection when not required.
- Care was taken to check for any infinite loop that exists in code before executing the code.

7.3 TESTING METHODS

While Box Testing

Also known as glass box, structural, clear box and open box testing. A software testing technique whereby explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike black box testing, white boxtesting uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do; it means that he must be completely aware that for particular input a particular output must be obtained. The main benefit of this type of testing is Tester can see if the program diverges from its intended goal. This test concentrates on the examination of the code rather than the specification. We have included three different forms of white box testing.

Statement Coverage Criterion:

This is the simplest coverage criterion. We are checking in it that each statement ofthe program was executed “at least once”.

Branch Coverage Criterion:

An improvement over statement is **Branch Coverage**. In that we are running a series Of test to ensure that all branches are tested at least once.

Path Coverage Criterion:

There are many errors which were not detected by statement or branch testing. The reason is that some errors are related to some combination of branches and it may not check in other test. We are checking in this test if all paths of programs are executed or not.

Black Box Testing

Black-box and white-box are test design methods. Black-box test design treats the system as a "black-box", so it doesn't explicitly use knowledge of the internal structure. Black-box test design is usually described as focusing on testing functional requirements. Also known as behavioral, functional, opaque-box and closed-box.
Black Box Testing was helpful us to find error such as:

1. Interface error
2. Incorrect or missing functions.
3. Errors in data structures or external database access.
4. Performance Errors.

Unit Testing

Unit testing is a method of testing the correctness of a particular module of source code. The idea is to write test cases for every non-trivial function or method in the module so that each test case is separate from the others if possible. The developers mostly do this type of testing. In this method of testing we test all individual components to ensure that they operate correctly. Each component is tested independently without other system components.

Integration Testing

It is the phase of software testing in which individual software modules are combined and tested as a group. It follows unit testing and precedes system testing. The purpose of Integration testing is to verify functional, performance and reliability requirements placed on major design items.

It takes as its input modules that have been checked out by unit testing, groups them in larger aggregates, applies tests defined in an Integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.

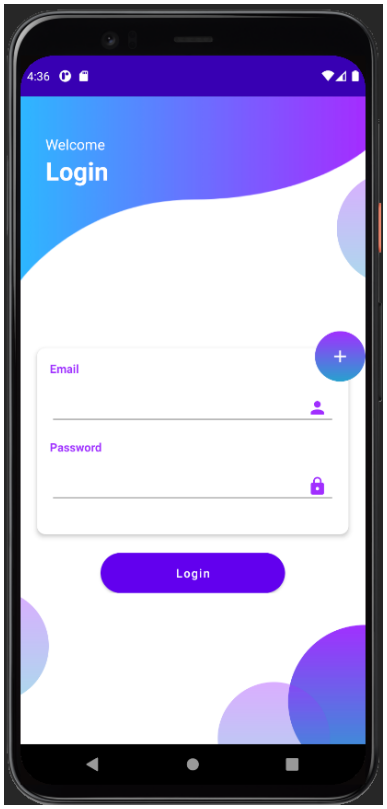
Interface Testing
The implemented system is user friendly and does not require any extra knowledgefor using the application and hence it passes the interface testing

7.4 Test Cases

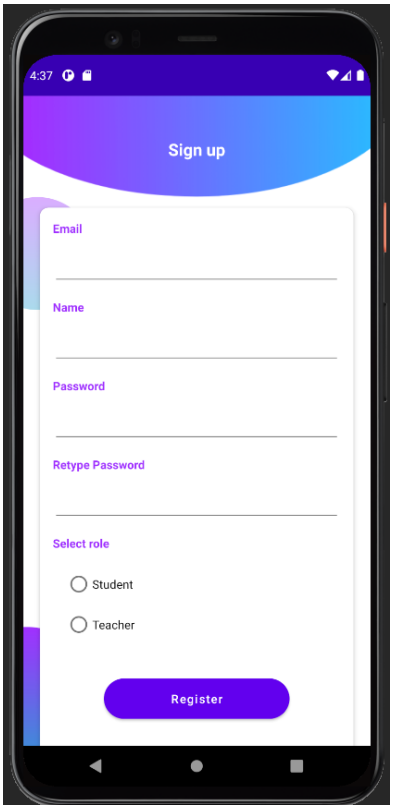
ENTITY	TEST CASE	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
Sign-up	Validation	Successfully Register and redirect to log in Page	Successfully register and redirect to log in page	pass
Login	Validation	Redirect to Home page	Redirect to Home page	pass
Sign-up As Teacher	Validation	Redirect to Home page of Teacher	Redirect to Home page of	pass
Create Classroom	Enter details	Create a new Classroom with a Unique joining code	Create a new Classroom with a Unique joining code	pass
View Students Attendance	View details	Display Attendance details Of students for specified Subject	Display Attendance details of students for specified Subject	pass
Organise Quiz	Enter details	Display the entered quiz data with respected options	Display the entered quiz data with respected options	pass
Mark Attendance	Generate and Scan QR	Mark Present in database	Mark Present in database	pass
Performance Review	View Review	Display given review to student	Display given review to student	pass

8. USER MANUAL

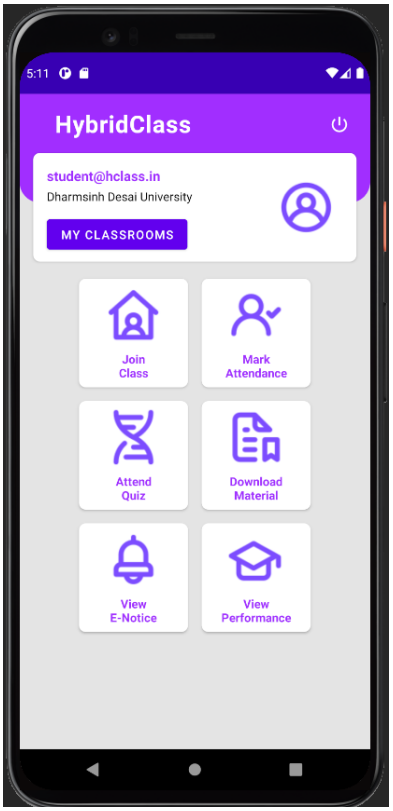
1) Login Page



2) Sign up

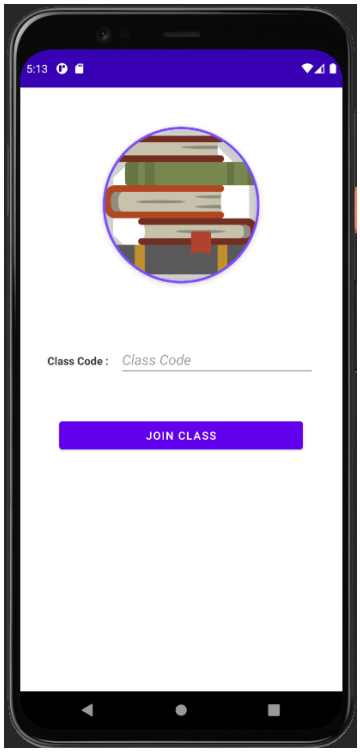


3) Student Dashboard

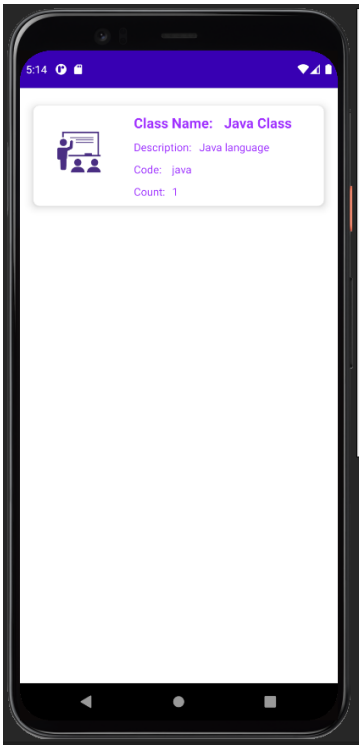


4) Join Class

5) My Class

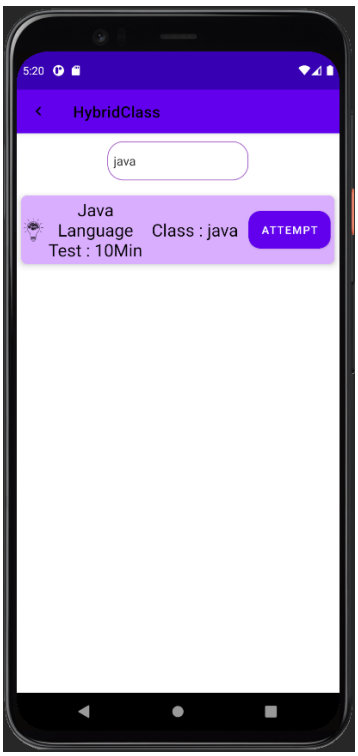


5) Student List

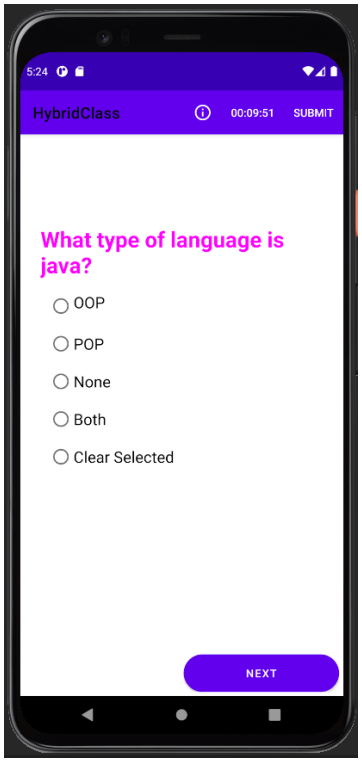


6) Attend Quiz

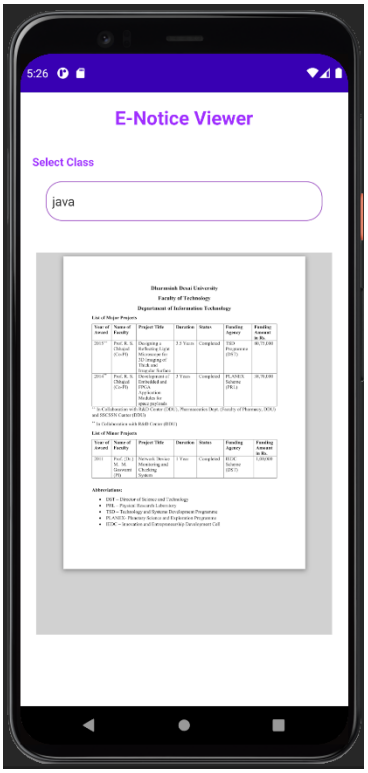
7) Attempt Quiz



8) View E-Notice



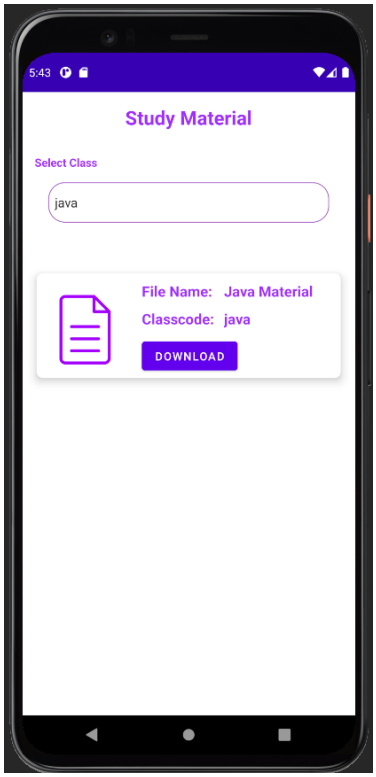
9) Student List



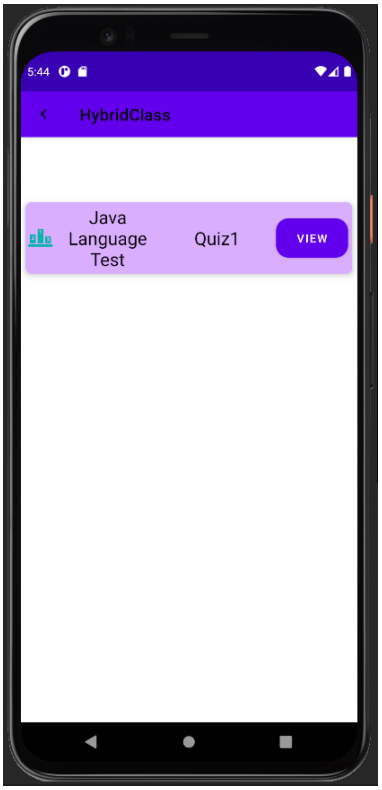
10) Attend Quiz

11) View Material

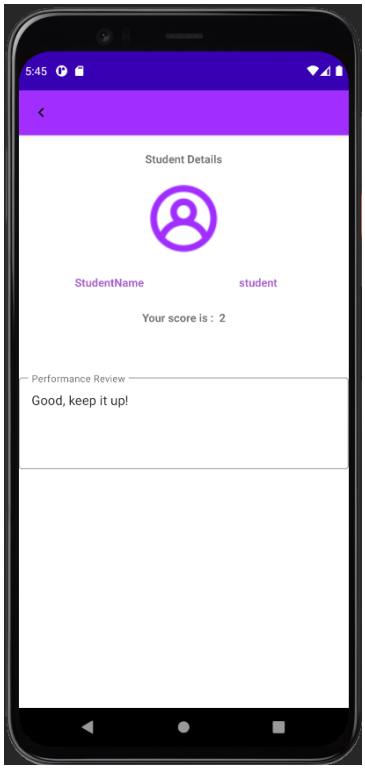
12) View Performance



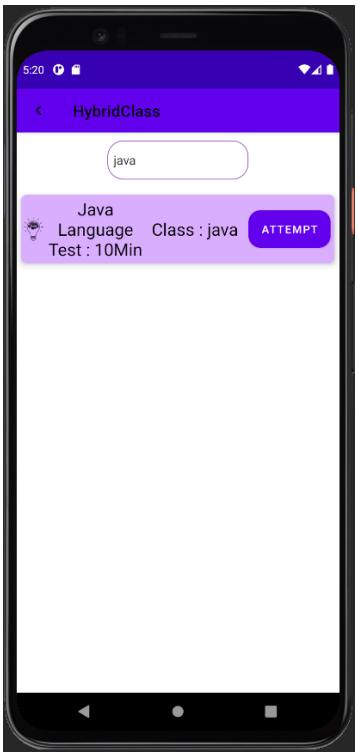
13) View Review



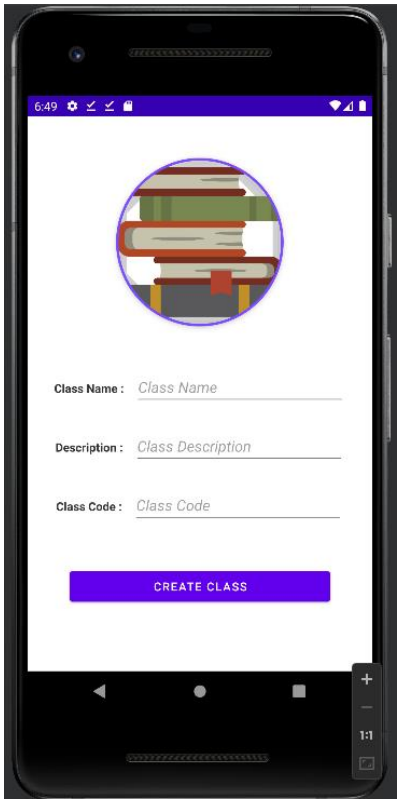
14) Attend Quiz



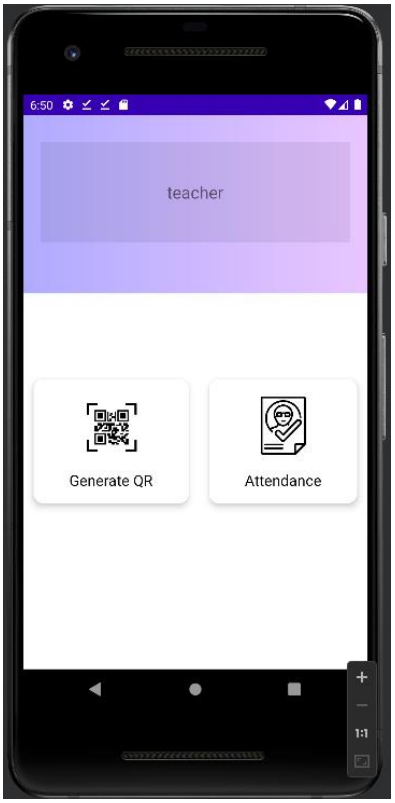
15) Create Class



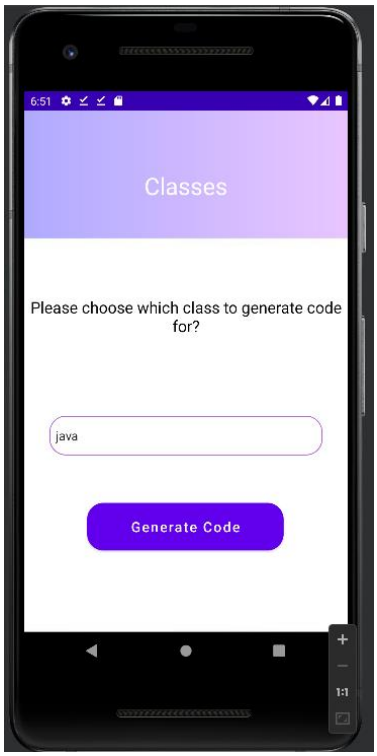
16) Create Attendance



17) Generate QR

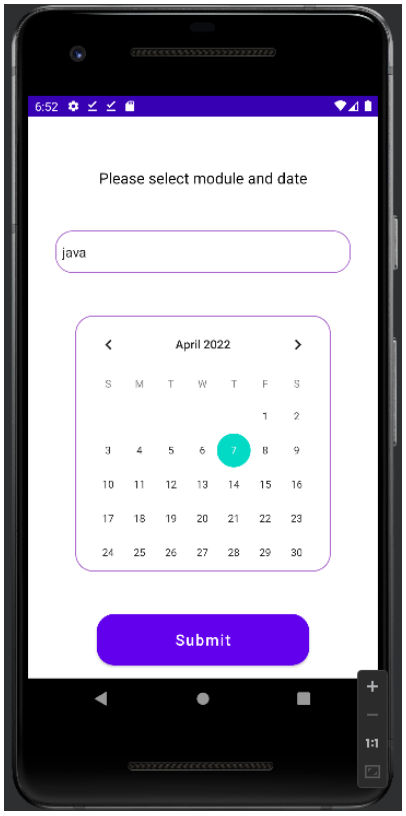


18) QR Code

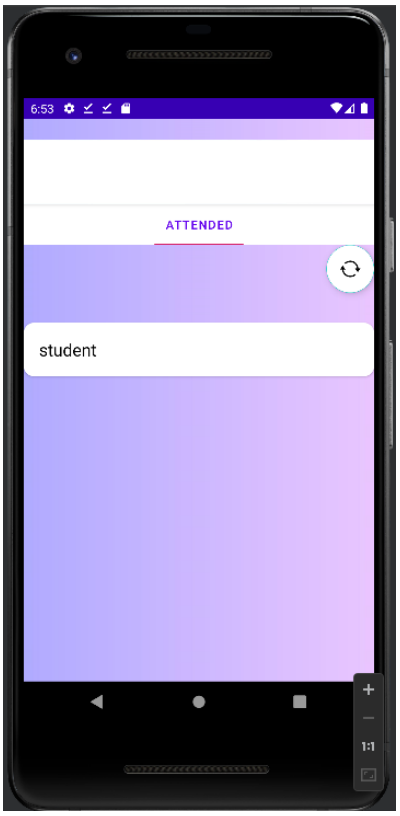


19) View Attendance

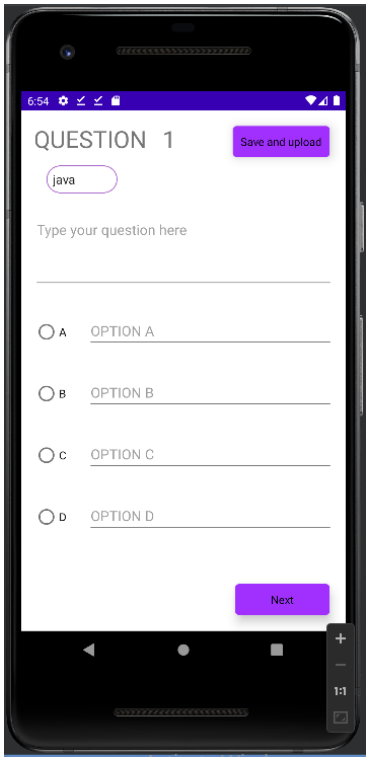
20) View Attendance



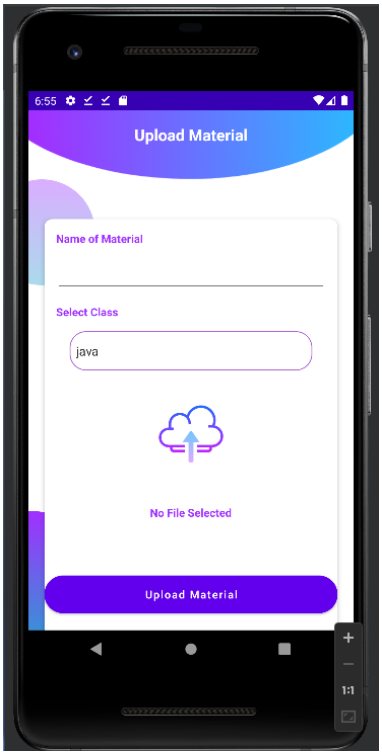
21) Create Quiz



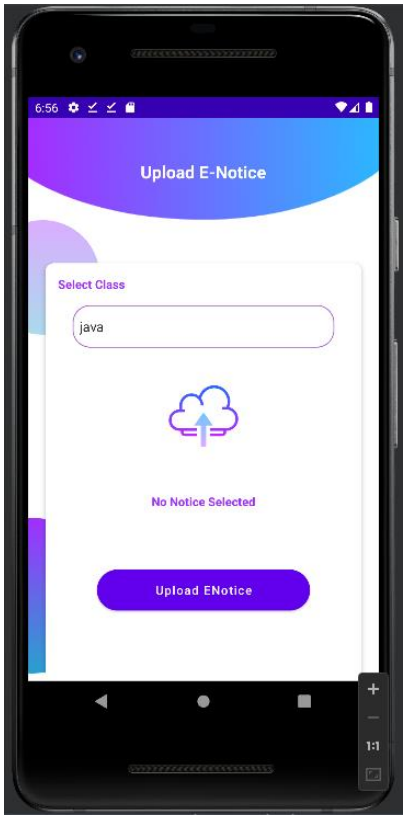
22) Give Material



23) Upload Notice

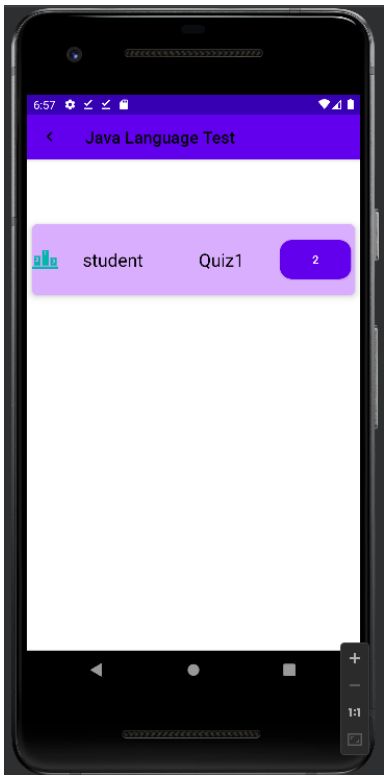


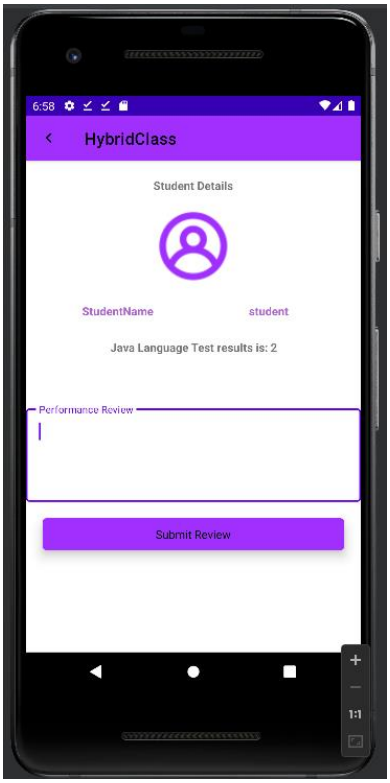
24) View Performance Review



25) View Quiz Attempters

26) Give Review





9. LIMITATION AND FUTURE ENHANCEMENT

9.1 LIMITATIONS

- Any student can login to other student's account and mark attendance.
- The Internet must be on the whole time while using the application.

9.2 FUTURE ENHANCEMENTS

- GUI could be enhanced in future.
- Functionality to report any incorrect data in the application.
- Our application's start-up and loading could be enhanced.

10. CONCLUSION AND DISCUSSION

10.1 CONCLUSION

We provide 1 step solution for all the requirements related to a Classroom. We have tried to keep the UI alluring and easy enough for our users to feel comfortable. There would be obvious limitation in users as our application would require internet connection.

10.2 DISCUSSION

10.2.1 Self-Analysis of Project Viabilities

In our opinion, this project has served the goal that we set when we started. It provides a platform for Teacher and Students the benefits of a physical classroom with the convince of a 'no-physical-bar'. Availability of study materials and other video lectures were made easy via the application. Student can choose courses, attend lectures, take quiz, view their attendance record, progress report, etc as per their convenience and Teachers can provides a means of collaborative learning for the students through this application.

10.2.2 Problem Encountered and Possible Solutions

There were so many problems encountered during this project: -

1. Problem to maintain databases and change them often.
2. Need to change some functionality fully which leads to doing the whole work again.
3. Many instances where the errors weren't understandable.
4. The entire database had to be rebuild thrice because of system errors

10.2.3 Summary of project work

We have completed our project work using software engineering and system analysis and design approach. We have done work with preplanned scheduling in accordance with the given time and produced the result-oriented progress in project development.

REFERENCES

https://www.stackoverflow.com/
https://firebase.google.com/
https://www.draw.io
https://developer.android.com
https://www.youtube.com/freecodecamp