



IT2010 – Mobile Application Development

BSc (Hons) in Information Technology

2nd Year

Faculty of Computing

SLIIT

2023 – Assessment 02

Description	Student ID	Name
Group Leader	IT21156960	Kodithuwakku C.K
Member 2	IT21164330	Arandara S.D.
Member 3	IT21166174	Navindi R.L.S
Member 4	IT21184444	Sumanasekara W.H.U.

Application Topic	Online education application
Group Name	Code warriors

Project Declaration

We, the members of Code Warriors, hereby declare that our group project is entirely authentic and original. We have conducted thorough research and analysis to ensure that our work is not plagiarized or copied from any other sources.

We have followed all guidelines provided by our LIC and have complied with all ethical and academic standards.

We take full responsibility for the authenticity of our work and understand the implications of academic dishonesty. We have worked collaboratively to produce this project, and each member has contributed to the best of their abilities.

We hereby affirm that our project represents our honest effort and commitment to academic integrity, and we take pride in presenting it as our own.

Charith

Group Leader (Signature)
<>Group Leader Name>>

Contents

Introduction	4
Git Repository	5
Contribution.....	6
1. IT21156960 (Kodithuwakku C.K) – Class management and Progress tracker	6
2. IT21164330 (Arandara S.D.) – Quiz management	11
3. IT21166174 (Navindi R.L.S) – Book Store management	15
4. IT21184444 (Sumanasekara W.H.U.) – Blog management.....	19

Introduction

As a result of current unstable situation of politics and economy, Sri Lanka's education sector is experiencing significant issues.

Many colleges and institutions struggle to provide basic services and facilities to the students due to lack of funding. As a result, education has declined in quality, and many pupils are finding it difficult to keep up with their studies. So, to overcome this problem, our developing team has come up with the idea of an app called "EduX" to help the students in our country.

With the help of various innovative tools, the "EduX" app provides students with all the educational support they need to advance their knowledge. This mobile application involves three principal parties. They are students, teachers, and donors. Students can enroll for courses and quizzes. The service that the teachers are offering is free, but if any donors would like to contribute money for it, they can do so through this app. Donating money isn't compulsory here. Donors who want to give books for the benefit of students can do so through this approach. Users now have the option to create and publish blog posts directly from their mobile device. We have made it easy for users to write and design their blog entries, including photographs and videos, and to preview their work before publishing.

Our team's ultimate objective in creating this educational application was to give users a fun and engaging way to learn even in the midst of a serious economic crisis in the nation, as well as to give them hope for their future aspirations.

Git Repository:

<https://github.com/chamithZ/EduAppdemo>

The screenshot shows the GitHub repository page for `chamithZ/EduAppdemo`. The repository is public and has 3 branches and 0 tags. The main branch contains 18 commits from `bad1b77` 5 hours ago. The commits are listed as follows:

File	Commit Message	Time Ago
<code>.idea</code>	updatedCv3	5 hours ago
<code>app</code>	updatedCv3	5 hours ago
<code>gradle/wrapper</code>	first commit	5 days ago
<code>.gitignore</code>	first commit	5 days ago
<code>build.gradle</code>	first commit	5 days ago
<code>gradle.properties</code>	first commit	5 days ago
<code>gradlew</code>	first commit	5 days ago
<code>gradlew.bat</code>	first commit	5 days ago
<code>settings.gradle</code>	first commit	5 days ago

On the right side, there is an **About** section with the message "No description, website, or topics provided." It also shows 0 stars, 1 watching, 0 forks, and a link to "Report repository". The **Releases** section indicates "No releases published" and a link to "Create a new release". The **Packages** section shows "No packages published" and a link to "Publish your first package". The **Contributors** section lists two contributors: `chamithZ` (Chamith Kavinda) and `SehanArandara` (Sehan Arandara).

Contribution

1. IT21156960 (Kodithuwakku C.K)- Class management and Progress tracker

Class management and Progress tracker

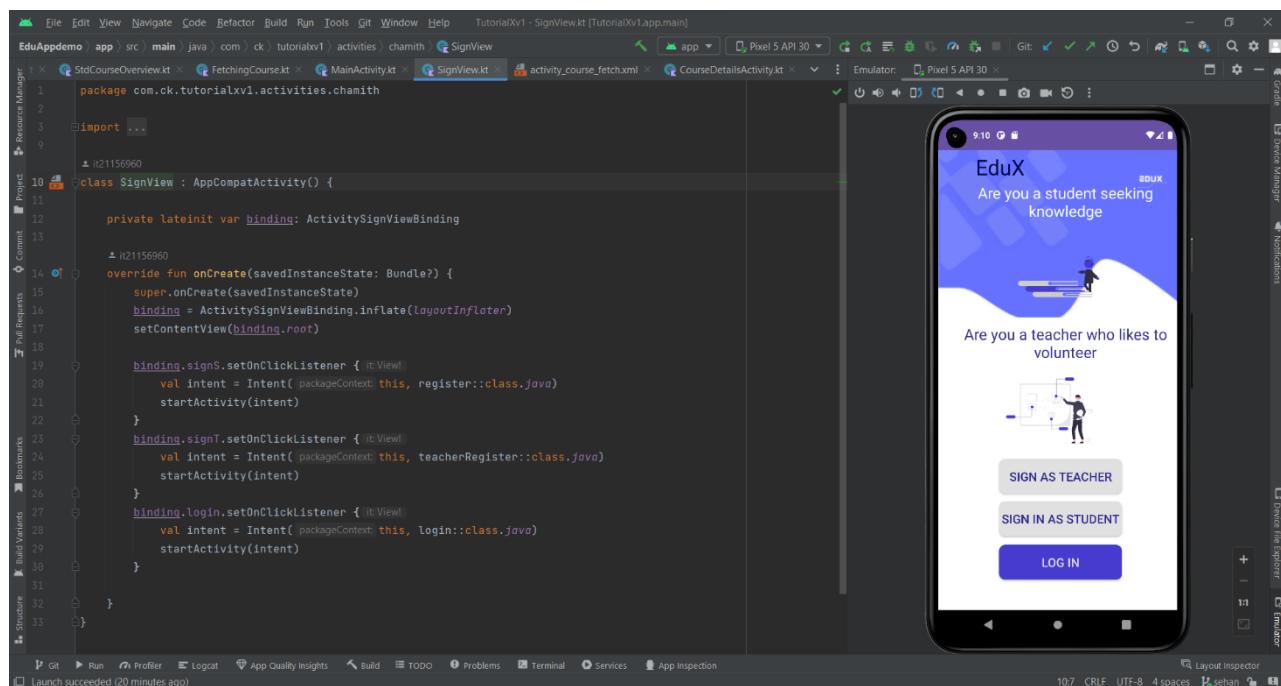
As an individual responsible for managing courses and users in this project, I was tasked with providing a user-friendly interface for students to register and view courses. I was also responsible for enabling students to enroll in courses and view their enrolled courses in a dedicated "My Courses" section. If a student wished to unenroll from a course, they could do so easily.

The enrollment feature was designed to incentivize students to continue learning by awarding points for each course they enrolled in. The number of points awarded depended on the user's level. These points were then used to calculate the user's progress, which was displayed in their profile. The progress tracking feature was effective in motivating students to continue learning and achieving their goals.

In addition to managing students, I was responsible for enabling teachers to register and add classes to the system. The system provided teachers with an easy-to-use interface to add classes with subject, grade, time, and date details. Teachers could also view their courses and update or delete them if required.

The course management features proved to be valuable additions to the system. They allowed students to easily find and enroll in courses and enabled teachers to manage their classes effectively. Additionally, the points-based enrollment system incentivized students to continue learning and improved their overall performance. The progress tracking feature provided students with valuable feedback on their progress and motivated them to continue learning.

Signup :



The screenshot shows the Android Studio interface with the TeacherRegister.kt file open in the code editor. The code implements an AppCompatActivity for teacher registration, utilizing DataBindingUtil for view inflation and FirebaseAuth for authentication. The right side of the screen displays a Pixel 5 API 30 emulator running the application, showing a registration form titled 'Join as Teacher' with fields for name, email address, contact number, age, password, re-type password, qualification, and bank AC number. A checkbox for agreeing to terms and conditions is also present, along with a 'Register Now' button.

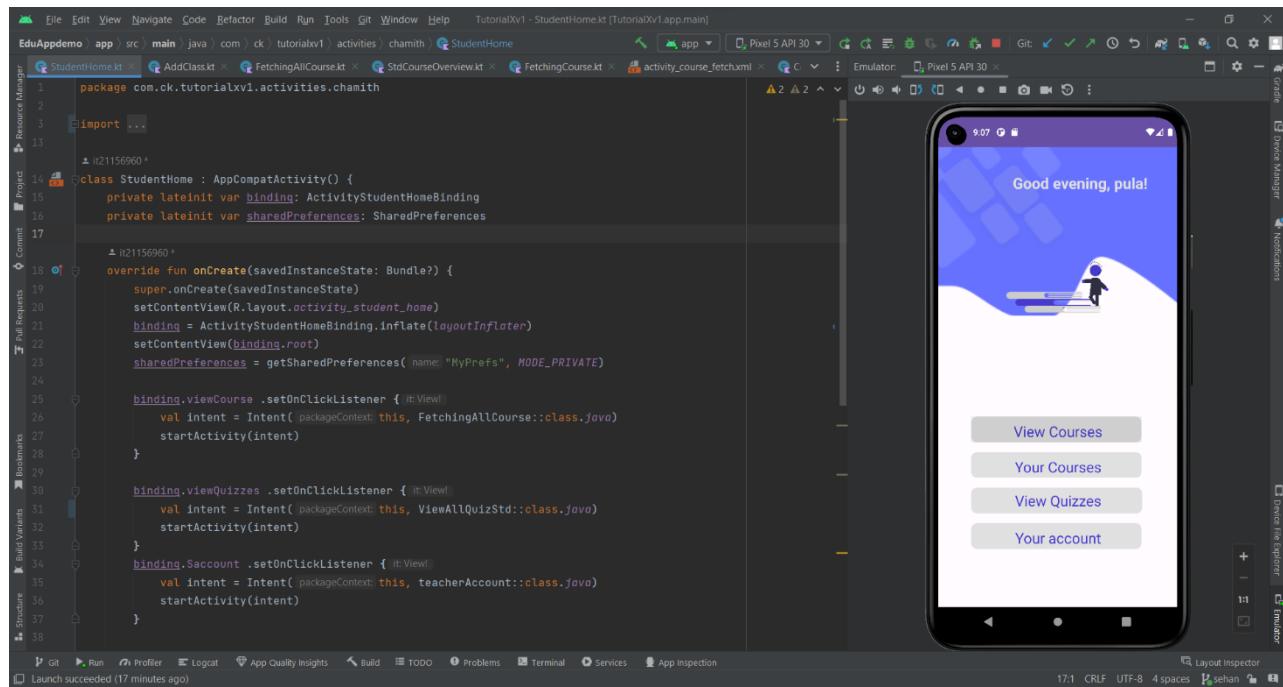
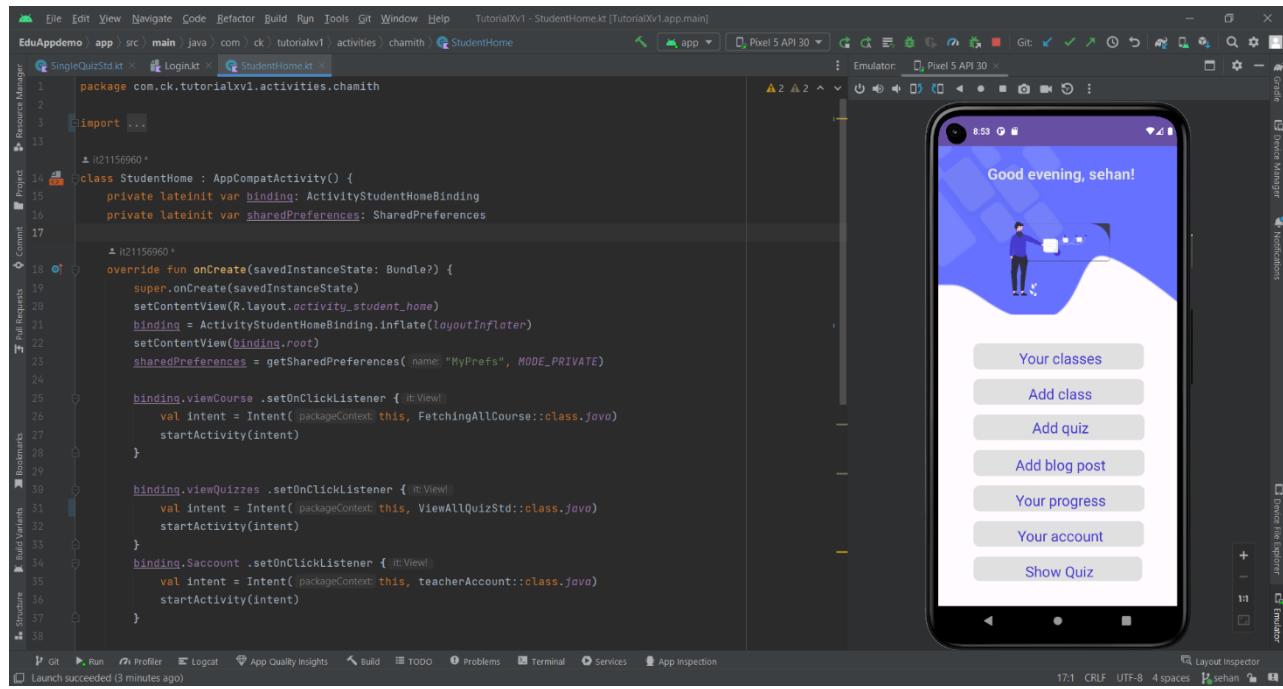
```
1 package com.ck.tutorialxv1.activities
2
3 import ...
4
5
6 class teacherRegister : AppCompatActivity() {
7
8     private lateinit var binding: ActivityTeacherRegisterBinding
9     private lateinit var firebaseAuth: FirebaseAuth
10
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14
15         binding = ActivityTeacherRegisterBinding.inflate(layoutInflater)
16         setContentView(binding.root)
17         firebaseAuth= FirebaseAuth.getInstance()
18
19
20         binding.Tregister.setOnClickListener{ it:View// register
21             val name = binding.Tname.text.toString()
22             val email = binding.Temail.text.toString()
23             val pass = binding.Tpassword.text.toString()
24             val repass = binding.TrePassword.text.toString()
25             val contactnum = binding.Tphone.text.toString()
26             val age = binding.Tage.text.toString()
27             val qualification = binding.qualification.text.toString()
28             val bankAc= binding.bankAc.text.toString()
29
30         }
31
32
33
34
35
36
37 }
```

Login :

The screenshot shows the Android Studio interface with the Login.kt file open in the code editor. The code implements an AppCompatActivity for user login, using ActivityLoginBinding for view inflation and FirebaseAuth for authentication. It includes logic to check if the user is already logged in via SharedPreferences and to start the Main Activity if true. The right side of the screen displays a Pixel 5 API 30 emulator running the application, showing a login screen with fields for email and password, a 'forgot password?' link, and a 'LOG IN' button. A keyboard is visible at the bottom of the emulator screen.

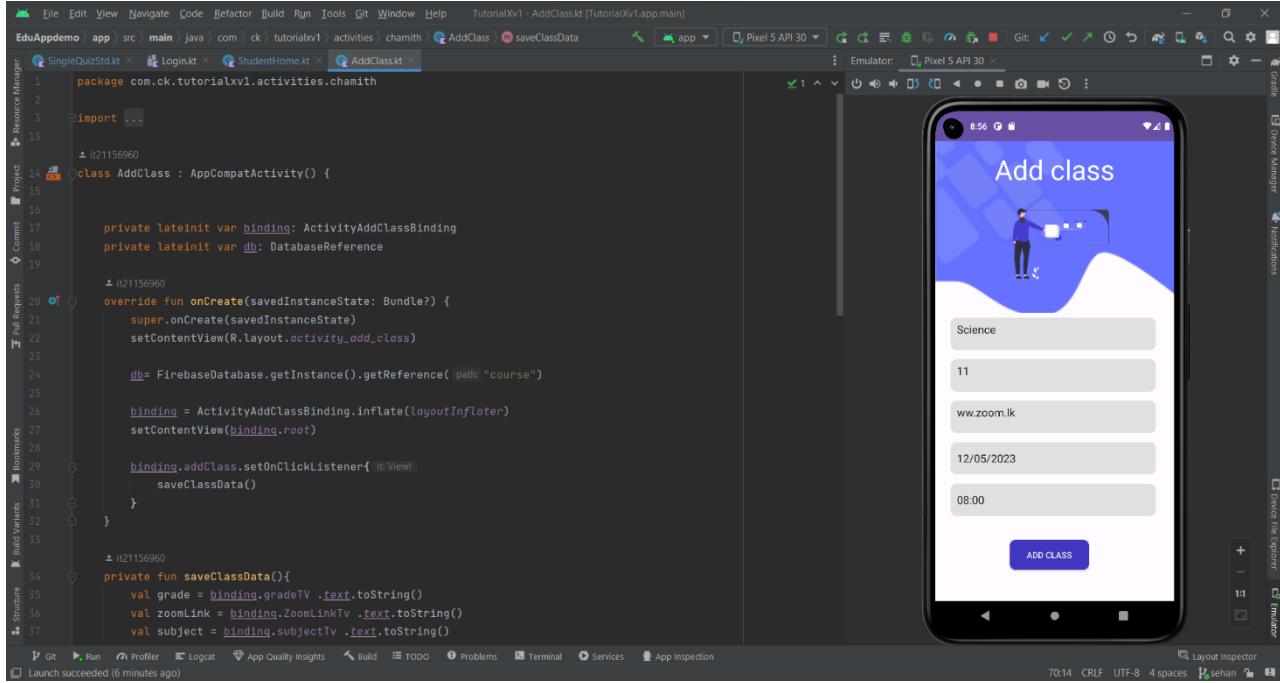
```
1 package com.ck.tutorialxv1.activities.chamith
2
3 import ...
4
5
6 class login : AppCompatActivity() {
7
8     private lateinit var binding: ActivityLoginBinding
9     private lateinit var firebaseAuth: FirebaseAuth
10    private lateinit var btnSave: Button
11    private lateinit var sharedpreferences: SharedPreferences
12
13
14    override fun onCreate(savedInstanceState: Bundle?) {
15        super.onCreate(savedInstanceState)
16        binding=ActivityLoginBinding.inflate(layoutInflater)
17        setContentView(binding.root)
18        btnSave=findViewById(R.id.login2)
19        firebaseAuth= FirebaseAuth.getInstance()
20        sharedpreferences = getSharedPreferences( name: "MyPrefs", Context.MODE_PRIVATE)
21
22
23        binding.textView4.setOnClickListener { it:View//
24            val intent= Intent( packageContext: this, SignView::class.java)
25            startActivity(intent)
26        }
27
28
29        if (sharedpreferences.getBoolean("isLoggedIn", false)) {
30            // If the user is already logged in, navigate to MainActivity
31            val grade = sharedpreferences.getString("grade", null)
32        }
33
34
35
36
37
38
39
40 }
```

Home Page

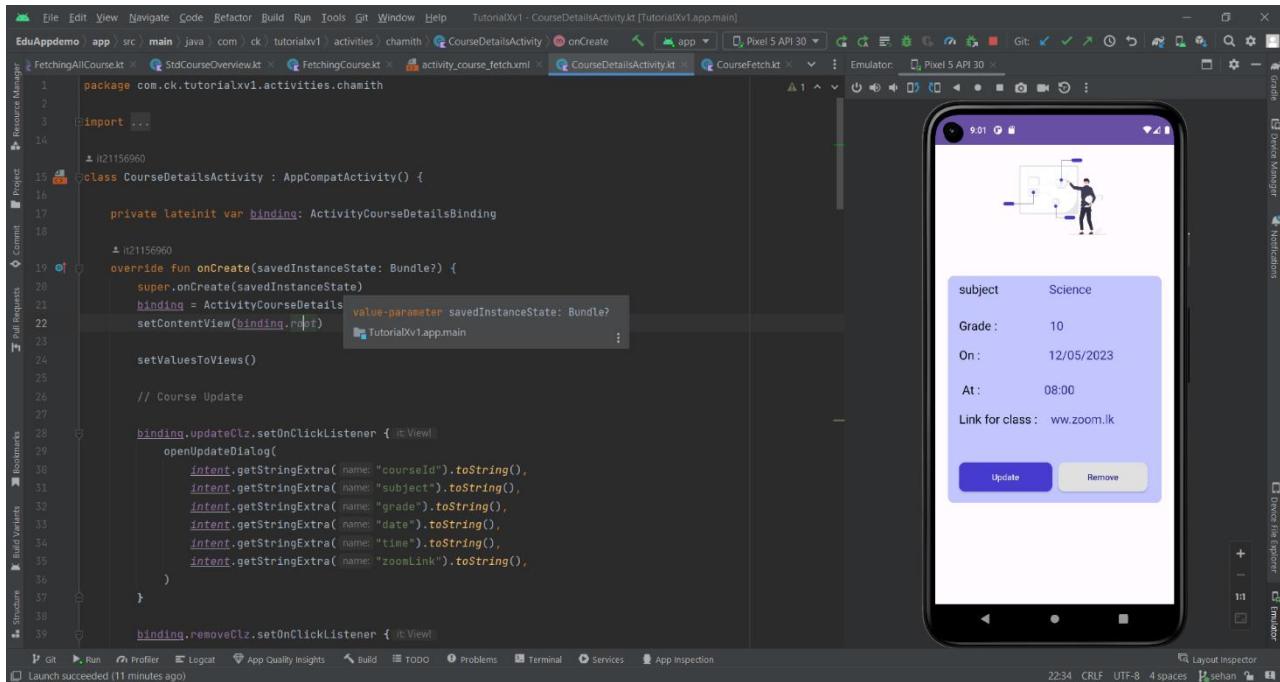


CRUD OPERATIONS

1. Create Operation



2. Read Operation



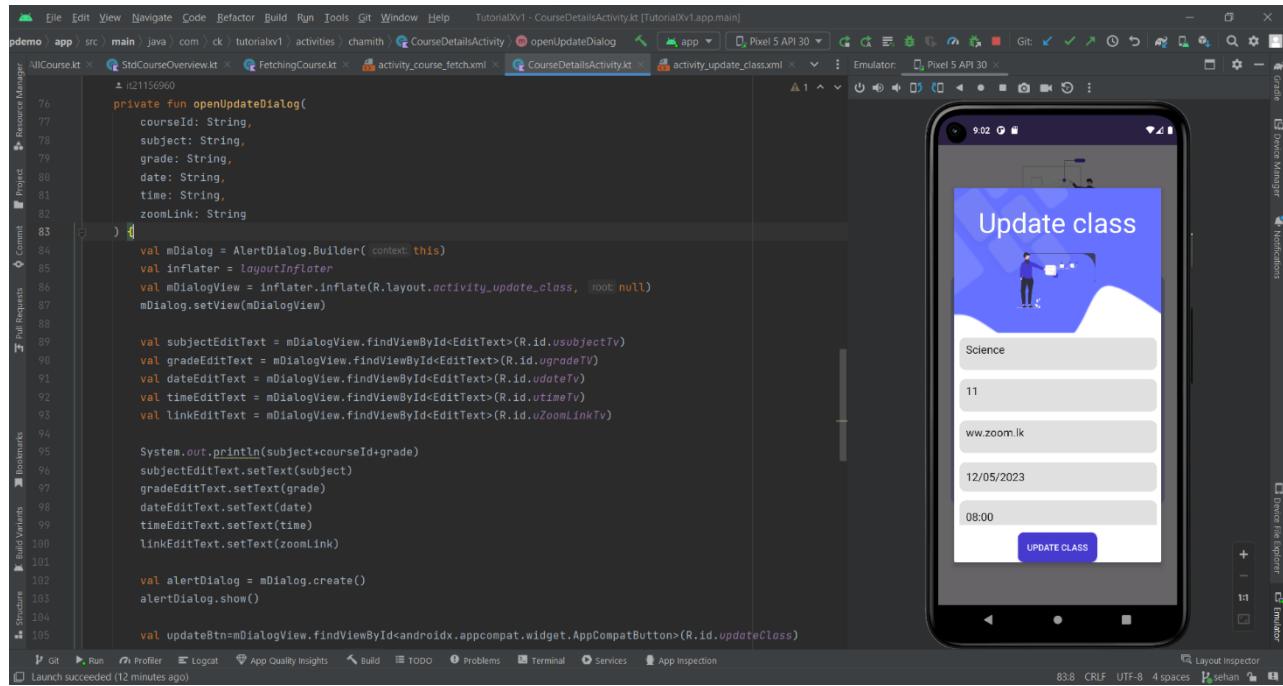
The screenshot shows the Android Studio interface with the following details:

- File menu:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Toolbar:** TutorialXv1 - FetchingAllCourse.kt [TutorialXv1.app.main].
- Project Structure:** EduAppdemo > app > src > main > java > com > ck > tutorialxv1 > activities > chamith > FetchingAllCourse.kt.
- Code Editor:** The FetchingAllCourse.kt file is open, showing Java code for an AppCompatActivity. It includes imports for Context, AppCompatActivity, ActivityCourseFetchBinding, ArrayList, courseModel, DatabaseReference, FirebaseAuth, and FirebaseFirestore. The code initializes variables for courseList, binding, db, auth, and userId. It overrides onCreate to set the layout and initialize Firebase. It also defines a getCourseData() method to handle course data retrieval.
- Emulator:** A Pixel 5 API 30 emulator is running, displaying a simple UI with the text "Science 11".
- Side Panels:** Resource Manager, Project, Commit, Full Requests, Bookmarks, Build Variants, Structure, and Layout Inspector.
- Bottom Bar:** Git, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, and App Inspection.
- Log:** Launch succeeded (8 minutes ago).

The screenshot shows the Android Studio interface with the following details:

- File menu:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Project navigation:** EduAppdemo (app), src/main/java/com/ck/tutorialxv1/activities/chamith/FetchingCourse.kt (selected).
- Code editor:** The FetchingCourse.kt file contains Java code for an AppCompatActivity. It includes imports for Context, AppCompatActivity, ActivityCourseFetchBinding, ArrayList, courseModel, DatabaseReference, FirebaseAuth, and FirebaseFirestore. The onCreate method initializes a binding object, sets the content view, and retrieves course data from Firebase. A getCourseData() method is also present.
- Emulator:** A Pixel 5 API 30 emulator is running, displaying a course list screen. The screen shows "History 11" and "Science 11".
- Toolbars and panels:** Top bar includes icons for Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, and App Inspection. Bottom bar includes icons for Git, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, and App Inspection. Bottom status bar shows "Launch succeeded (18 minutes ago)" and "Pixel 5 API 30".

3. Edit Operation



The screenshot shows the Android Studio interface with the code editor open. The code is for an 'update' operation, specifically for updating course details. It includes logic to inflate a dialog, set its view, and handle various edit texts for subject, grade, date, time, and zoom link. The code also prints the concatenated course ID and grade to the console. A preview of the 'Update class' dialog is shown on the right, featuring fields for subject (Science), grade (11), zoom link (www.zoom.lk), date (12/05/2023), and time (08:00). A blue 'UPDATE CLASS' button is at the bottom.

```
private fun openUpdateDialog() {
    courseId: String,
    subject: String,
    grade: String,
    date: String,
    time: String,
    zoomLink: String
}

val mDialog = AlertDialog.Builder(context).create()
val inflater = LayoutInflater.from(context)
val mDialogView = inflater.inflate(R.layout.activity_update_class, null)
mDialog.setView(mDialogView)

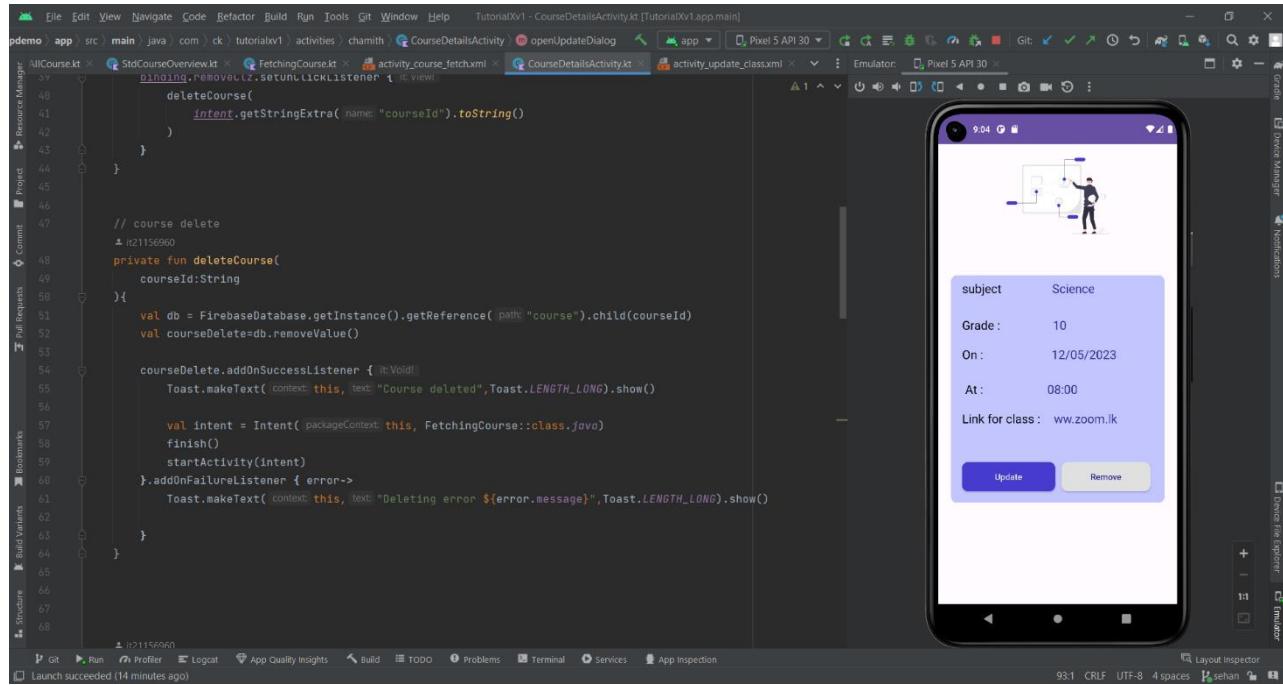
val subjectEditText = mDialogView.findViewById<EditText>(R.id.usubjectTv)
val gradeEditText = mDialogView.findViewById<EditText>(R.id.ugradeTv)
val dateEditText = mDialogView.findViewById<EditText>(R.id.udateTv)
val timeEditText = mDialogView.findViewById<EditText>(R.id.utimeTv)
val linkEditText = mDialogView.findViewById<EditText>(R.id.ZoomLinkTv)

System.out.println(subject+courseId+grade)
subjectEditText.setText(subject)
gradeEditText.setText(grade)
dateEditText.setText(date)
timeEditText.setText(time)
linkEditText.setText(zoomLink)

val alertDialog = mDialog.create()
alertDialog.show()

val updateBtn = mDialogView.findViewById<Button>(R.id.updateClass)
```

4. Delete Operation



The screenshot shows the Android Studio interface with the code editor open. The code is for a 'delete' operation, specifically for deleting a course. It includes logic to remove a course from the database and handle the deletion process. A preview of a course details dialog is shown on the right, listing the subject (Science), grade (10), date (12/05/2023), time (08:00), and zoom link (www.zoom.lk). It also includes 'Update' and 'Remove' buttons.

```
private fun deleteCourse(intent: Intent) {
    val courseId = intent.getStringExtra("courseId")
    FirebaseDatabase.getInstance().getReference("course").child(courseId).removeValue()
}

private fun deleteCourse(courseId: String) {
    val db = FirebaseDatabase.getInstance().getReference("course").child(courseId)
    val courseDelete = db.removeValue()

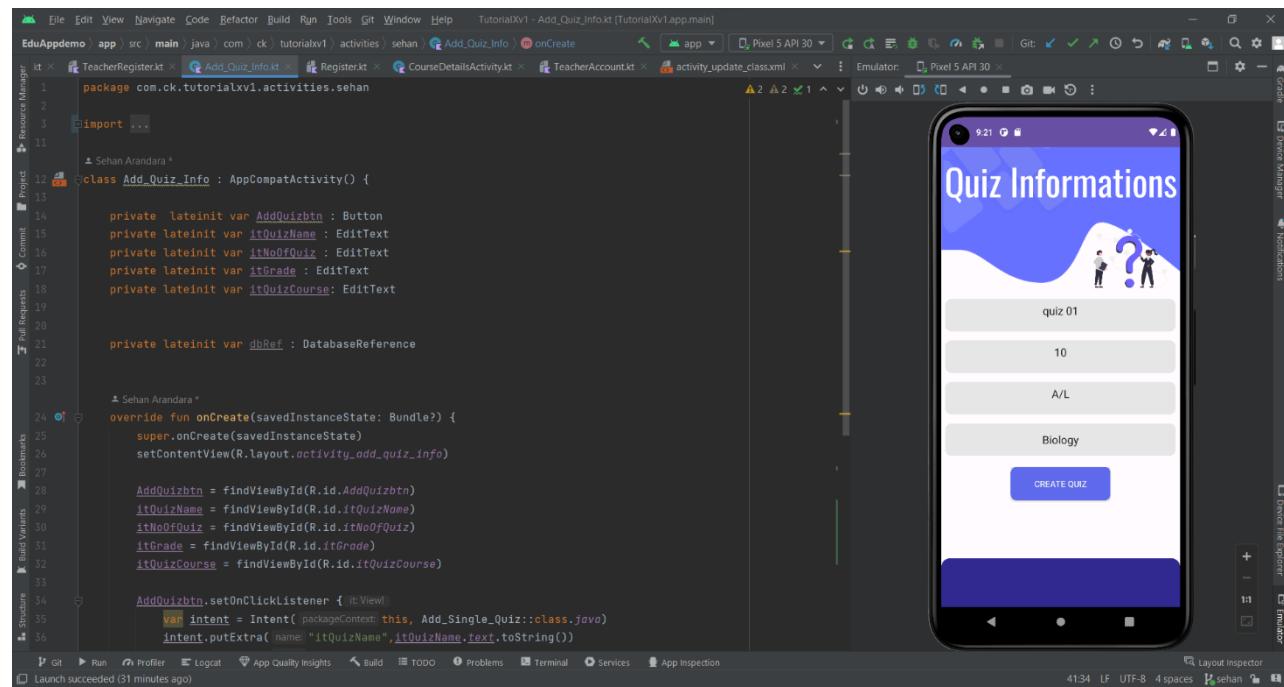
    courseDelete.addOnSuccessListener { }
        .addOnFailureListener { error ->
            Toast.makeText(context, "Deleting error ${error.message}", Toast.LENGTH_LONG).show()
        }
}
```

2. IT21164330 (Arandara S.D.) - Quiz management

I am the one who responsible for implement the quiz section in our app . In Our EduX app aims to provide free education to student , so that we are implementing quizzes . It is a vey good way to test their knowledge in various subjects . Teacher can create quizzes by providing the necessary information about the quiz . teacher can edit or delete the quizzes as they want. Both teachers and student can view the quizzes and student can do the quizzes . I will implement some function to calculate the marks and the grade of the student after the quiz finished.

CRUD OPERATIONS

1. Create Operation



The screenshot shows the Android Studio interface with the code editor open to `Add_Single_Quiz.kt`. The code defines a class `Add_Single_Quiz` extending `AppCompatActivity`. It initializes various UI components like buttons, edit texts, and a database reference. The preview window shows a mobile application with a purple header and a white content area titled "Sample Question 1". It displays four options (Op 1, Op 2, Op 3, Op 4) and a bottom bar with buttons for "3", "NEXT", and "Done".

```
package com.ck.tutorialxv1.activities.sehan

import ...

class Add_Single_Quiz : AppCompatActivity() {

    private lateinit var nextBtn : Button
    private lateinit var doneBtn : Button
    private lateinit var itQuizName : String
    private lateinit var itNoOfQuiz : String
    private lateinit var itGrade : String
    private lateinit var itQuizCourse : String
    private lateinit var iquestion : EditText
    private lateinit var incorrectOp : EditText
    private lateinit var op01 : EditText
    private lateinit var op02 : EditText
    private lateinit var op03 : EditText
    private lateinit var op04 : EditText
    private var quizList = mutableListOf<Question>()
    private lateinit var dbRef : DatabaseReference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_add_single_quiz)

        dbRef = FirebaseDatabase.getInstance().getReference( path: "Quizzes" )

        nextBtn = findViewById(R.id.nextBtn)
    }
}
```

2. Read Operation

The screenshot shows the Android Studio interface with the code editor open to `QuizResultActivity.kt`. The code defines a class `QuizResultActivity` extending `AppCompatActivity`. It initializes a `QuizResult` TextView. The preview window shows a mobile application with a purple header and a white content area titled "Results". It displays the text "Your Marks is" followed by "0%" and a "Exit" button.

```
package com.ck.tutorialxv1.activities.sehan

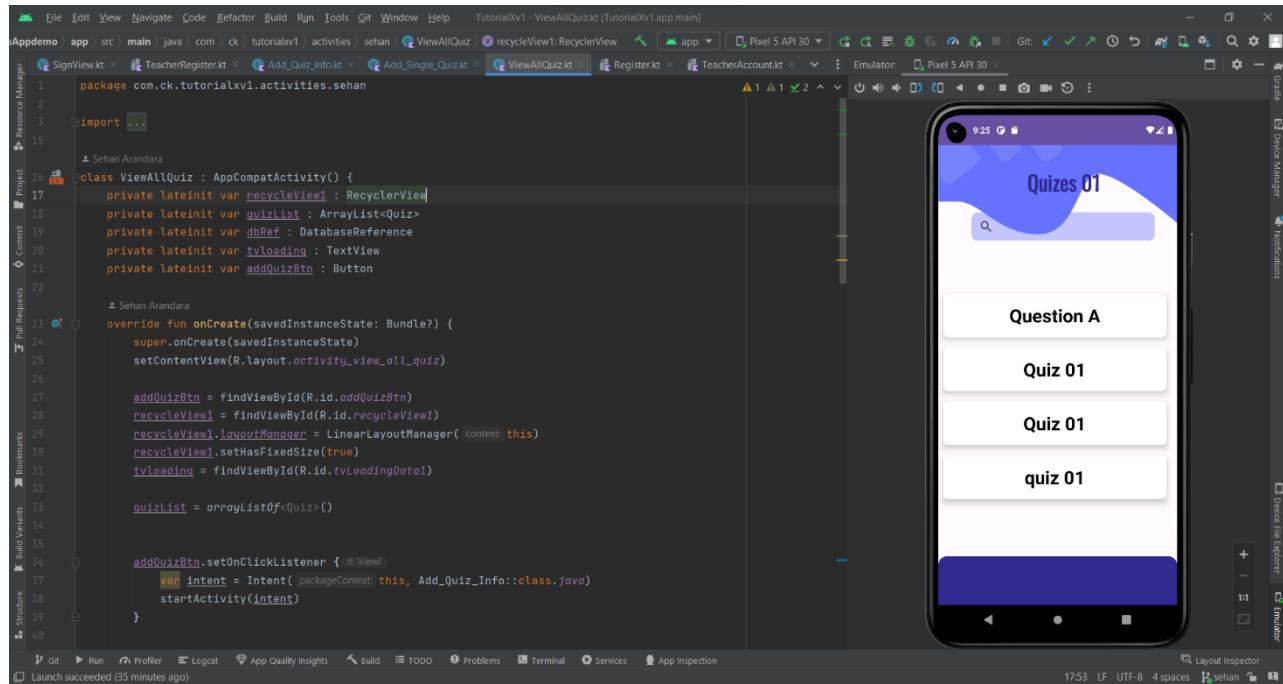
import ...

class QuizResultActivity : AppCompatActivity() {

    private lateinit var QuizResult : TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_quiz_result)

        QuizResult = findViewById(R.id.QuiZResult)
        QuizResult.text = intent.getStringExtra( name: "result" )
    }
}
```



The screenshot shows the Android Studio interface with the code editor open to the `SingleQuizStd.kt` file. The code defines a class `SingleQuizStd` that extends `AppCompatActivity`. It initializes variables for a quiz, question text view, and four option buttons. It also sets up a database reference and handles the creation of the activity. The right side of the screen shows an emulator running an Android application with a purple-themed interface titled "Sample Question 1". It displays four options labeled "Op 1", "Op 2", "Op 3", and "Op 4", with a "NEXT" button at the bottom.

```
package com.ck.tutorialxv1.activities.sehan

import ...

class SingleQuizStd : AppCompatActivity() {

    private lateinit var quiz: Quiz
    private lateinit var questionTextView: TextView
    private lateinit var option1Button: Button
    private lateinit var option2Button: Button
    private lateinit var option3Button: Button
    private lateinit var option4Button: Button
    private lateinit var nextButton: Button
    private lateinit var dbRef: DatabaseReference
    private var currentQuestionNumber: Int = 1
    private var correctAnswerCount: Int = 0
    private var clickedAnswer: String = "0"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_single_quiz_std)

        val quizId = intent.getStringExtra(name: "id")

        dbRef = FirebaseDatabase.getInstance().getReference(path: "Quizes/$quizId")

        dbRef.addValueEventListener(object : ValueEventListener {
```

3. Update Operation

The screenshot shows the Android Studio interface with the code editor open to the `View_Quiz_Details.kt` file. The code contains an `onCreate` method that creates an alert dialog to update quiz information. It retrieves values from edit texts and updates a database record. The right side of the screen shows an emulator running an application with a title "Quiz Informations". A modal dialog titled "Updating Quiz 01 Record" is displayed, showing fields for "Quiz 01", "c", and "10", with a "Update Data" button.

```
private fun openUpdateDialog(id: String, title: String) {
    val mDialog = AlertDialog.Builder(context: this)
    val inflater = LayoutInflater
    val mDialogView = inflater.inflate(R.layout.activity_dialog_quiz_info, root: null)

    mDialog.setView(mDialogView)

    val etQuizName = mDialogView.findViewById<EditText>(R.id.etQuizName)
    val etQuizGrade = mDialogView.findViewById<EditText>(R.id.etQuizGrade)
    val etQuizCourse = mDialogView.findViewById<EditText>(R.id.etQuizCourse)
    val btnUpdateData = mDialogView.findViewById<Button>(R.id.btnUpdateData)

    etQuizName.setText(intent.getStringExtra(name: "title").toString())
    etQuizCourse.setText(intent.getStringExtra(name: "grade").toString())
    etQuizGrade.setText(intent.getStringExtra(name: "course").toString())

    mDialog.setTitle("Updating $title Record")

    val alertDialog = mDialog.create()
    alertDialog.show()

    btnUpdateData.setOnClickListener { v: View? ->
        updateQuizData(
            id,
            etQuizName.text.toString(),
            etQuizCourse.text.toString(),
            etQuizGrade.text.toString()
        )
    }
}

private fun updateQuizData(id: String, name: String, grade: String, course: String) {
    val dbRef = FirebaseDatabase.getInstance().getReference(path: "Quizes/$id")
    dbRef.child("name").setValue(name)
    dbRef.child("grade").setValue(grade)
    dbRef.child("course").setValue(course)
}
```

4. Delete Operation

The screenshot shows the Android Studio interface. On the left, the code editor displays Java code for a quiz deletion operation. On the right, a mobile device emulator shows the 'Quiz Informations' screen with quiz details and edit/delete buttons.

Java Code (View_Quiz_Details.kt):

```
47     btnDelete.setOnClickListener { view ->
48         deleteRecord(
49             intent.getStringExtra(name: "id").toString()
50         )
51     }
52
53 }
54
55 // Sehan Arandara
56 private fun deleteRecord(id: String) {
57     val dbRef = FirebaseDatabase.getInstance().getReference(path: "Quizes").child(id)
58     val mTask = dbRef.removeValue()
59
60     mTask.addOnSuccessListener { void: Void? ->
61         Toast.makeText(context: this, text: "Quiz is deleted", Toast.LENGTH_SHORT).show()
62         val intent = Intent(packageContext: this, ViewAllQuiz::class.java)
63         finish()
64         startActivity(intent)
65     }.addOnFailureListener { err: Exception? ->
66         Toast.makeText(context: this, text: "deleting error $err", Toast.LENGTH_SHORT).show()
67     }
68
69 // Sehan Arandara
70 private fun openUpdateDialog(id: String, title: String) {
71     val mDialog = AlertDialog.Builder(context: this)
72     val inflater = LayoutInflater
73     val mDialogView = inflater.inflate(R.layout.activity_dialog_quiz_info, root: null)
74
75     mDialog.setView(mDialogView)
76 }
```

Mobile Device Emulator Screen:

Quiz Informations

Quiz id	-NUkFkkcXoydLIBV7zn
Quiz Name	Quiz 01
Course	5
Grade	Science

Buttons: EDIT QUIZ, DELETE QUIZ

3. IT21166174 (Navindi R.L.S)- Book Store management

I am handling the Books management.

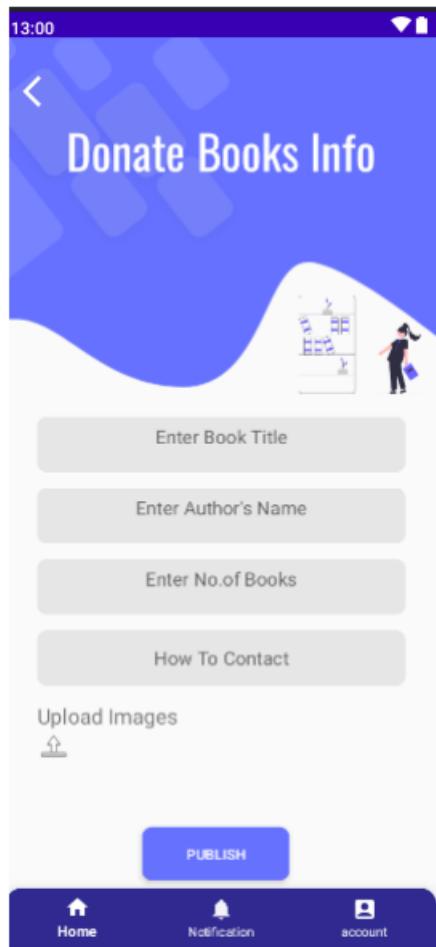
Principles

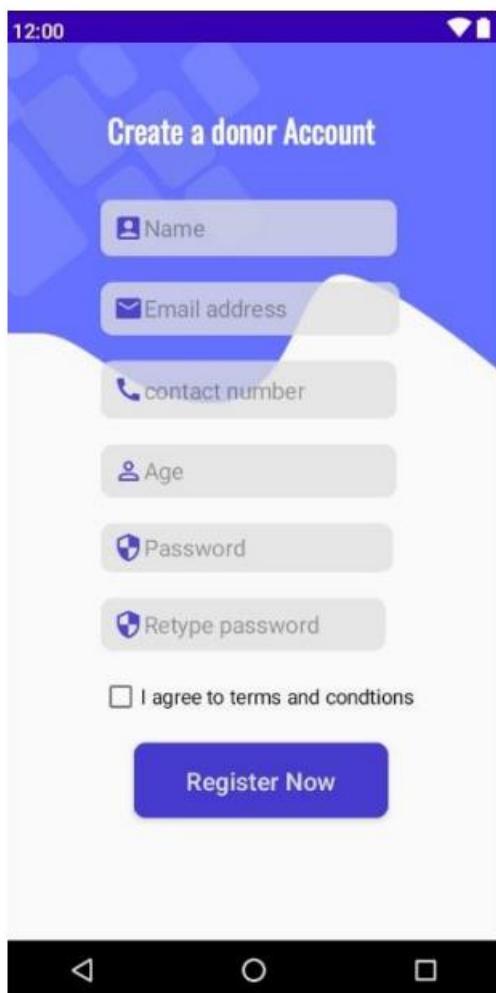
I have used be focused, be charming mobile mindset principles, and I used global guidelines principles such as responsiveness, and polish. Also, to get a good first impression I have used different buttons, images.

CRUD OPERATIONS

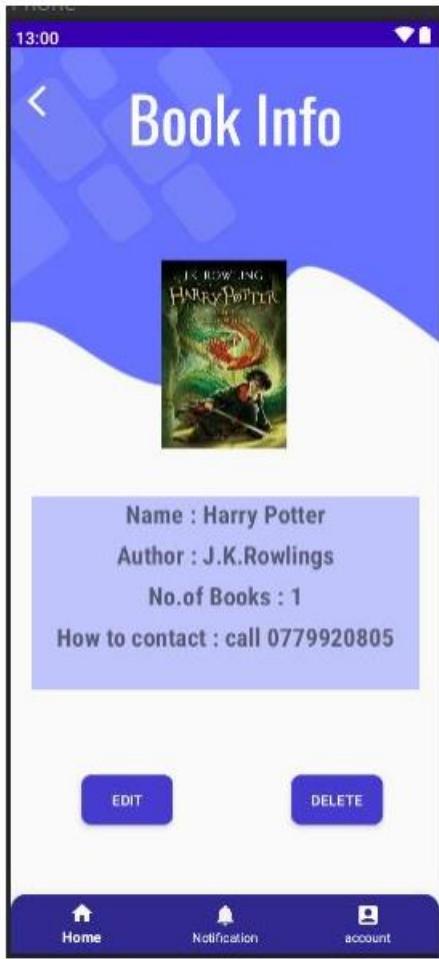
This page includes all the donated book information. The donated books are inserted and uploaded via this page. Users can view the book info as they required. Admin can update or delete any book using “Edit” and “Delete” buttons respectively if necessary. Only admin can edit and delete the details about books in this system. A user can retrieve the books that they have already published after publishing a new book using “Publish” button. And also donators can add books to the system and their contact details to donate their books physically. There is a calculation function used to show the book count that has been donated by a donator. All the data inserted are stored in Firebase database.

1. Create Operation





2. Read Operation



4. IT21184444 (Sumanasekara W.H.U.)– Blog management

In our EduX app I am the responsible for the blog management feature, which allows users to create, read, update, and delete (CRUD) blog posts. Users can create blog posts by filling in the topic and blog area. Once they submit their post, it will be saved to the app's database, and other users can see it. Other users can search for and read blog posts created by other users. They can also view their own blog posts and all the comments made about their own blog entries.

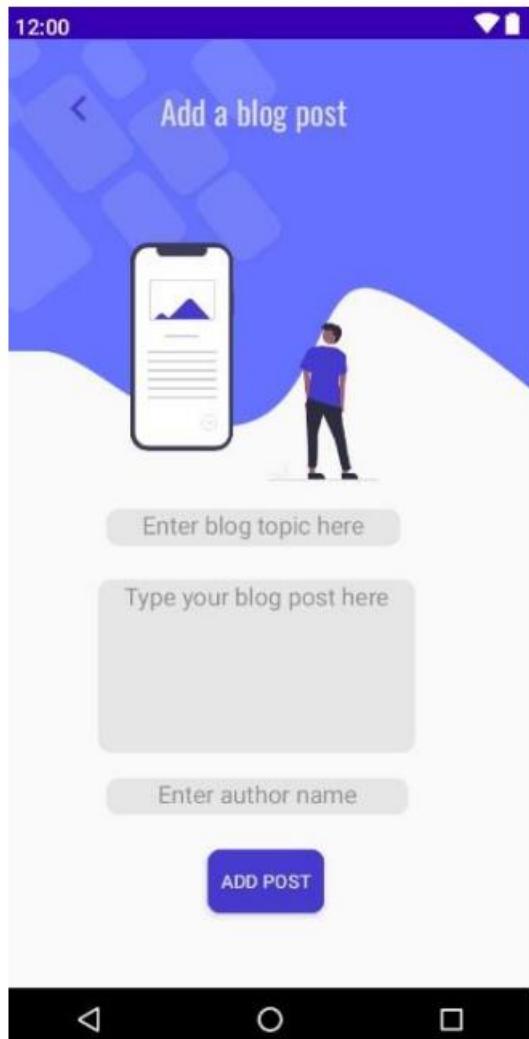
To provide more flexibility to the users, the app allows them to edit and delete their blog posts. They can edit the topic and blog area of their post and save the updated version. When a user no longer wants a particular blog post to be displayed on the app, they can delete it. The user can choose the post they wish to permanently remove from the database and then confirm their deletion choice. Additionally, users can add comments to other blog posts and edit their own comments as needed. This feature encourages interaction and feedback among users. To ensure the security and integrity of the app, appropriate permissions and access control will be

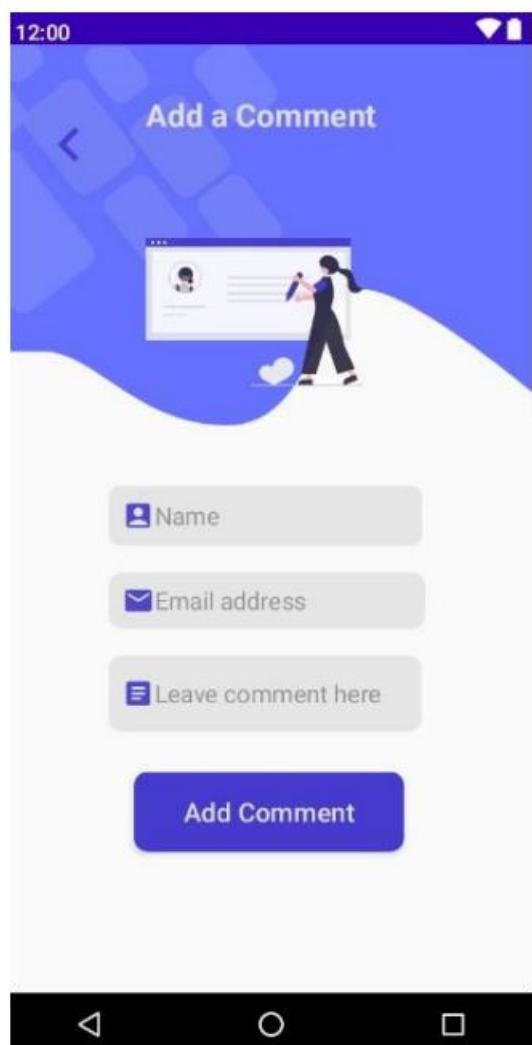
implemented. This will enable users to only edit and delete their own content and limit access to sensitive data to authorized personnel only.

Overall, these CRUD procedures give users the freedom to maintain their blog entries and interact with other users' content, encouraging interaction and sharing of knowledge within the app.

CRUD OPERATIONS

1. Create Operation





2. Read Operation

