UNIVERSITI MALAYA

WIA2007 MOBILE APPLICATION DEVELOPMENT

Semester 1, Session 2023/2024

Final Report

Lecturer: Dr. Chiew Thiam Kian

Occurrence: 5

Group: CTK1

Title: EduEmpower

NAME	MATRIC NUMBER
Cheok Yangding	S2199179/1
Bong Chee Hian	22004729/1
Chew Jia Hui	22004739/1
Goh Kah Kheng	22051683/1
Loh Pooi Hua	22004838/1
Teoh Zhi Yee	22058912/1

Table of Contents

1.0 Introduction

1.1 Project Background

In the current educational landscape, students and educators often face challenges in accessing high-quality and interactive learning resources tailored to their specific needs. The existing platforms may lack user-friendly interfaces and comprehensive features that cater to the diverse requirements of both secondary and tertiary education levels. Additionally, there is a pressing need to address the broader goal outlined in SDG 4, which emphasizes providing inclusive and equitable quality education, particularly focusing on equipping individuals with relevant skills for financial success.

Against this backdrop, the EduEmpower mobile application project emerges as a strategic response to these challenges. It seeks to fill the void in the educational technology sphere by developing a robust learning platform that not only addresses the immediate concerns of user-friendliness but also aligns with the broader objectives of SDG 4. The project envisions creating an inclusive space where students and educators can seamlessly engage in quality learning experiences, fostering a culture of lifelong learning and skill development.

1.1.1 Project Goal:

The project's goal is to develop the EduEmpower mobile application, strategically aligned with SDG 4 (Quality Education) and, more specifically, Target 4.4, to increase the number of individuals equipped with essential skills for financial success.

1. **Affordable and Accessible Courses:**
   - Develop and provide cost-effective, accessible courses tailored for students at the secondary and tertiary education levels.
   - Ensure inclusivity by removing barriers to education, making quality learning resources available to a broad audience.
2. **Personalized Learning Experience:**
   - Implement features within the EduEmpower app that offer personalized course recommendations based on individual learning preferences and goals.
   - Foster a culture of lifelong learning by tailoring the educational journey to the unique needs and aspirations of each learner.

1.1.2 Project Scope:

The project's scope emphasizes the development of a fully functional prototype for the EduEmpower mobile application, with a dedicated focus on aligning with SDG 4 (Quality Education). This initiative aims to be realized within a stringent three-month timeframe,

emphasizing the creation of an interactive and user-friendly learning platform tailored for the distinct needs of students and educators at both secondary and tertiary education levels.

1. **Prototype Development:**
   - Develop and deliver a functional prototype of the EduEmpower mobile application within a precise three-month timeline.
   - Ensure that the prototype incorporates key features essential for fostering quality education, aligning with SDG 4's mission of inclusive and equitable learning opportunities.
2. **Comprehensive Features:**
   - Implement vital features such as seamless user registration, intuitive course selection interfaces, interactive quizzes, flashcards for enhanced learning experiences, dynamic forum discussions, and user profiles.
   - Each feature is meticulously designed to contribute to a holistic and high-quality learning environment, directly addressing the objectives of SDG 4 by focusing on inclusivity, accessibility, and relevance in education.
3. **User-Centric Approach:**
   - Prioritize a user-friendly design, emphasizing accessibility and ease of use for both students and educators.
   - Tailor the EduEmpower app to cater to the unique requirements of learners and facilitators at the secondary and tertiary education levels, ensuring an engaging and enriching educational experience.
4. **Alignment with SDG 4:**
   - Emphasize the SDG 4 goal of providing inclusive and equitable quality education by developing features that bridge educational gaps and address the diverse needs of students and educators.
   - Continuously evaluate the prototype's development to ensure alignment with SDG 4's Target 4.4, aiming to increase the number of individuals equipped with relevant skills for financial success through accessible and quality education.

1.2 Functional Requirements

Module (Epic)	Backlog ID	Functional requirement	Description
A: Class Material Module	A001	Study and manage class material	Enable students to explore captivating class materials designed in a story mode format to maintain their engagement and interest.
	A002	Progress tracker	To allow students and instructors to track students' progress on their learning.
	A003	Manage class material	To allow instructors to upload class materials in each course and edit the sequence and content of class materials to be displayed to the students.
B: Course Module	B001	Browse courses	To allow students to browse a diverse selection of available courses that were prepared by instructors.
	B003	Manage courses	To allow instructors to add, edit and manage their courses, and see statistics of the course like number of people enrolled and completed the course
	B006	Course recommendation	To allow students to view their dashboard/home page with suggested relevant courses based on their field of interest as collected upon account creation
C: Flashcard Module	C001	Browse and manage flashcards	To allow students to browse flashcards using active recall and to allow instructors to upload revision flashcards

	C002	Flashcard recommendation	To recommend students with flashcards to answer as revision using spaced repetition
	C003	Reward-based system for flashcards	To allow students to unlock flashcards using reward-based system after completing each topic
D: Quiz module	D001	Attempt quiz and review quiz	To allow students to attempt the quiz and review back for revision
	D002	Manage quiz	To allow instructor to add quiz for each topic
	D003	Analyze quiz	To allow instructors to analyze attempted quizzes for future enhancement.
E: Discussion Module	E001	Create discussion forum	To allow instructors to create a discussion forum for students to post their questions and exchange their opinions
	E002	Add comment	To allow end users to add comments regarding the questions in the discussion forum
	E003	Edit/delete comment	To allow end user to edit or delete their comments
F: Account Module	F001	Log in and register account	To allow students and instructors to register and login to their own account to save progress.
	F002	User settings	To allow end users to manage their account settings like changing their username, password and personal information.
	F003	Customer service/FAQ	To allow end users to reach out to customer service or view FAQ for app support.

1.3 Non-Functional Requirements

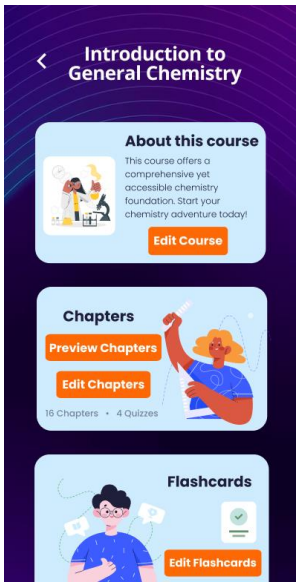
No	Non-functional Requirements	Descriptions
1	Response time	The application must exhibit efficient response times to ensure scalability. As the user base grows significantly, and a large audience engages with the app simultaneously, it should maintain quick response times to uphold a seamless user experience. The goal is to prevent any degradation in performance even during peak usage periods.
2	Security	To safeguard end users' data, stringent measures will be implemented to prevent any unauthorized access or data leaks. Robust security protocols will be in place to ensure the confidentiality and integrity of user information, adhering to industry standards and best practices in data protection.

2.0 App Design

2.1 User Interface Design Principles

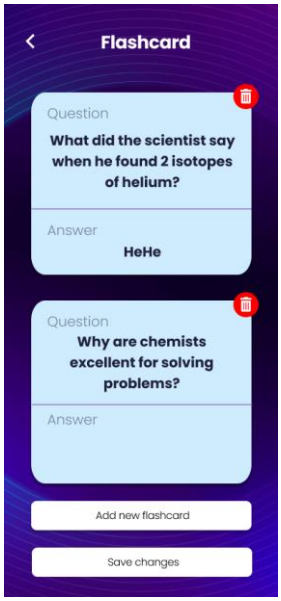
2.1.1 Colour Scheme

In defining our colour scheme, we opted for purple as the main colour, creating a thematic connection to the app's focus on education and study. To enhance visual appeal and create a balanced design, we introduced touches of blue and orange as contrasting elements. This intentional colour selection not only aligns with the app's purpose but also contributes to a harmonious and engaging user experience.



2.1.1 Layout and Typography

Card Layout



Our design incorporates a card-like layout, utilizing rounded rectangles to encapsulate content. This approach enhances user interaction by presenting information in a visually distinct and organized manner. Users can easily discern between different types of information, promoting clarity and a seamless navigation experience.

Consistent Margins and Alignment

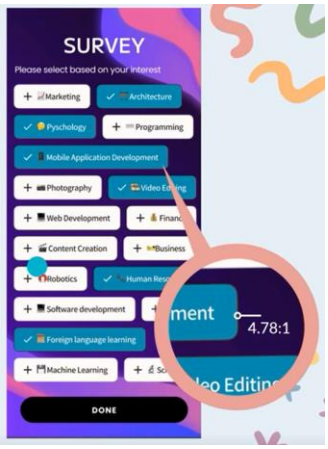
Throughout the app, we maintain a consistent approach to margins and alignment. Icons, for instance, are implemented with a padding larger than 48dp x 48dp. This intentional choice ensures a clean and organized layout, contributing to enhanced readability and navigability across various screens.

Readable Font Size

To prioritize readability, we utilize the "sp" unit for font size, allowing adaptability to the device's font settings. Furthermore, we ensure that the font size remains consistently above 12sp, guaranteeing that text content remains easily readable for end users.

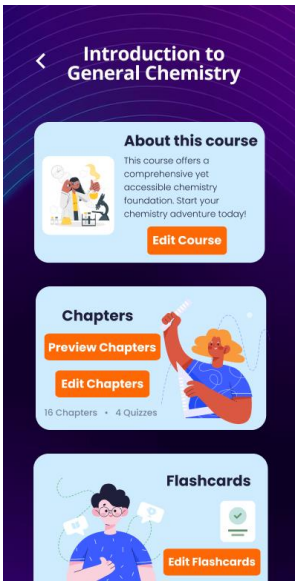
2.1.2 Colour and Contrast

Contrasted Button Design



Our approach to button design emphasizes contrast, with most buttons featuring bold colours that stand out from the surrounding UI. For instance, white buttons against a dark purple background create a visually distinctive and easily identifiable design. An example of this is evident in clustering buttons, where a contrast ratio of 4.78:1 is maintained between the container colour and the background colour.

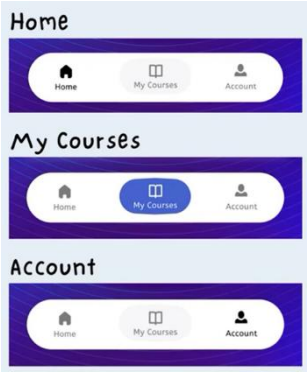
Interactive Elements



Interactive elements such as quizzes and flashcards employ a card-based design with contrasting colours. This intentional use of colour helps users differentiate interactive content from non-interactive elements, providing visual cues that enhance the overall user experience.

2.1.3 Feedback and Navigation

Visual Feedback



Incorporating visual feedback is a key aspect of our design philosophy. For example, when a user clicks a button, a change in color provides immediate visual feedback. This is notably implemented in icons within the bottom navigation, creating an intuitive and responsive user interface.

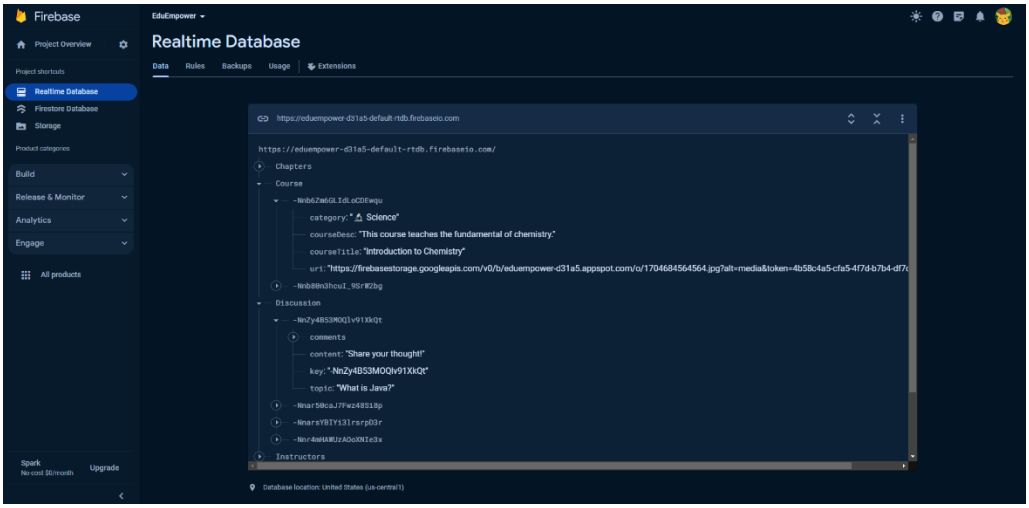
Consistent layout in navigation

To ensure a cohesive user journey, navigation elements are consistently positioned on every page, particularly within the bottom navigation. This consistent layout enhances user familiarity, simplifying navigation and contributing to an overall seamless user experience.

2.2 Data Storage and Design

2.2.1 Database

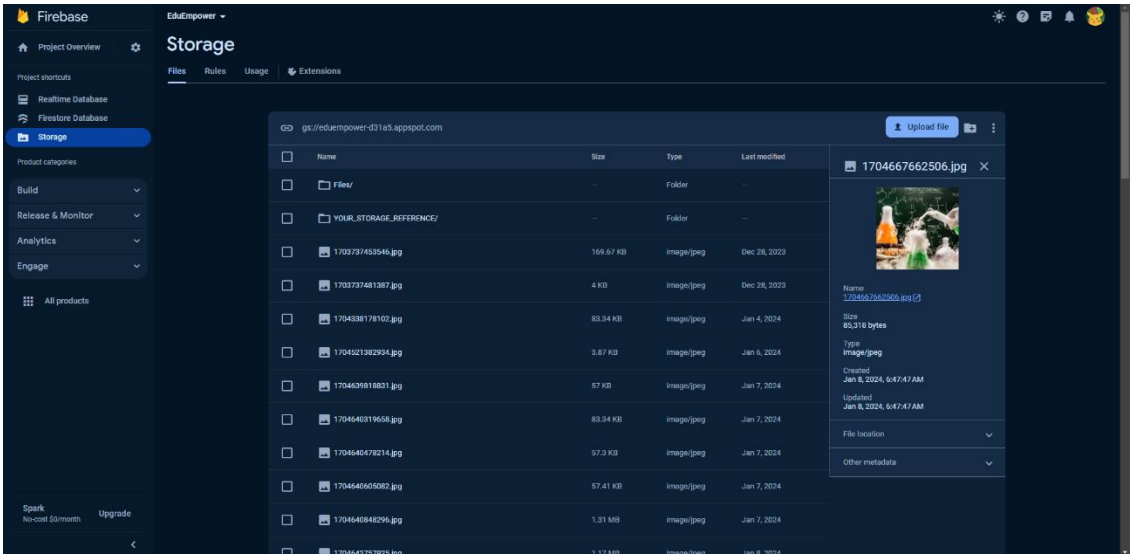
Firebase Realtime Database



In the context of EduEmpower, the selection of Firebase Realtime Database as our primary database solution is grounded in its NoSQL nature, offering flexibility and scalability. This choice aligns with the dynamic and diverse data requirements of the application. Firebase Realtime Database is adept at handling structured data in real-time, making it an ideal solution for managing user profiles, course details, and progress tracking. The NoSQL approach allows for efficient handling of unstructured data, adapting well to the evolving needs of the application as it scales.

2.2.2 Shared Storage

Firebase Cloud Storage



For shared storage, we opted for Firebase Cloud Storage to address the challenges associated with managing user-generated content, such as images, videos, and PDFs. Firebase Cloud Storage provides a scalable and secure platform for storing and retrieving large files. This solution is crucial for facilitating collaborative learning, allowing users to share and access multimedia content seamlessly. The integration of Firebase Realtime Database ensures real-time updates and synchronization, fostering a collaborative and dynamic educational environment. Firebase Cloud Storage aligns with our objective of creating a platform where users can contribute and access diverse learning materials.

2.2.3 Preferences

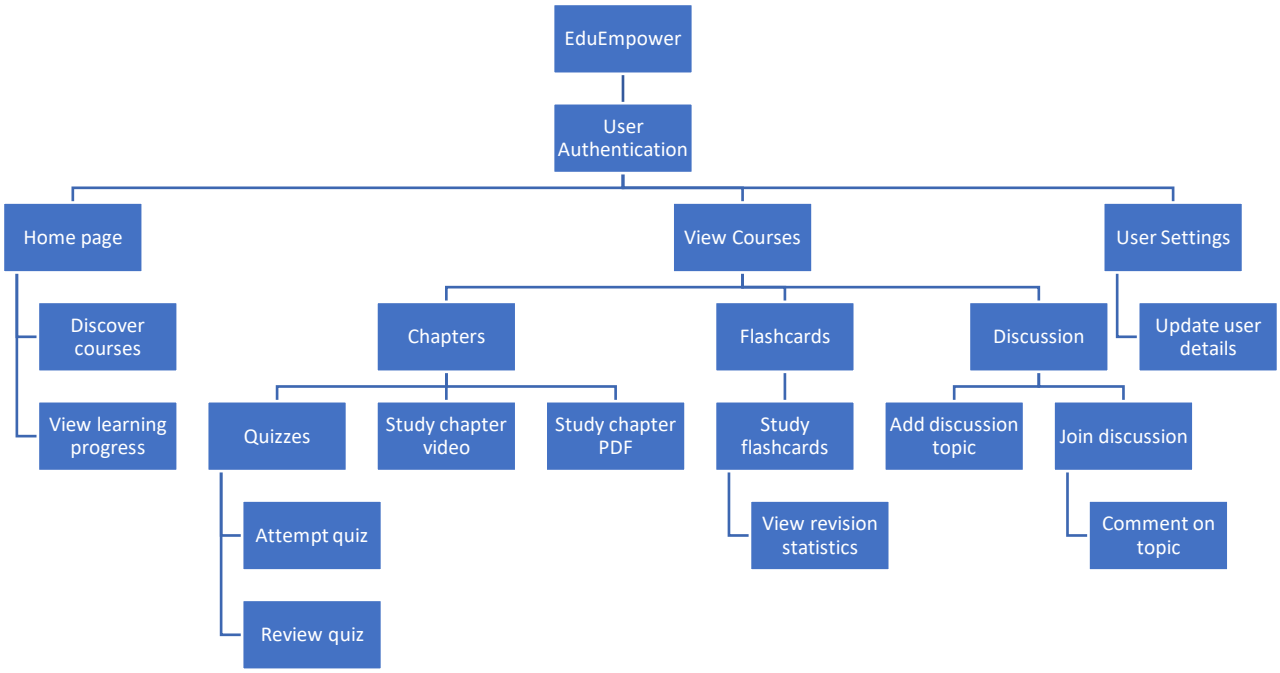
Shared Preferences

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

// Show DOB from database
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String dob = String.valueOf(snapshot.child( path: "DateOfBirth").getValue());
            editDOB.setText(dob);
        }
    }
});
```

To persist small amounts of app settings, user preferences, and essential data in a key-value pair format, we leverage SharedPreferences. This lightweight and efficient solution is particularly suitable for storing user-specific configurations across different sessions. In EduEmpower, SharedPreferences is employed to store the user ID, providing a quick and accessible means of identification across various modules and activities in the application. The simplicity and speed of SharedPreferences make it an optimal choice for managing lightweight data, enhancing the overall user experience by retaining personalized settings seamlessly.

3.0 App Flow

EduEmpower System Architecture Diagram



Introducing the System Architecture Diagram for the EduEmpower app, this visual representation provides an overview of the key features and functionalities that users can navigate within the application. The diagram outlines the educational journey, starting with the essential User Authentication process, which includes login and register functions to ensure secure access to the app.

Upon successful authentication, users are welcomed to the home page, where they encounter prominent features shaping their learning experience. The "Discover Courses" option beckons users to explore a diverse array of learning opportunities, while the "View Learning Progress" feature provides valuable insights into academic achievements and areas for improvement.

As users delve deeper into the app, they can explore specific courses, each structured with chapters housing various learning materials. This hierarchical arrangement includes engaging elements such as quizzes, study materials in video and PDF formats, interactive flashcards, and discussion forums. The "Attempt Quiz" and "Review Quiz" functionalities offer an immersive

assessment experience, while video and PDF resources cater to diverse learning preferences. Interactive flashcards, accompanied by detailed revision statistics, enhance the effectiveness of the learning process. Discussion forums empower users to actively participate in knowledge-sharing, enabling actions such as "Adding Discussion Topics," "Joining Discussions," and "Commenting on Topics."

In the user settings section, individuals can personalize their EduEmpower experience through the "Update User Details" feature, ensuring seamless management of user information for relevance and accuracy. This user-centric design emphasizes overall accessibility and adaptability, contributing to a positive and engaging learning environment. EduEmpower, with its structured and user-friendly interface, aspires to cultivate a community of lifelong learners by seamlessly integrating personalized experiences with a diverse set of educational tools and resources.

4.0 System Implementation

4.1 Development Environment Adopted by The Group

Android Studio with Java

In crafting EduEmpower, we strategically selected Android Studio with Java as our development environment, aligning with our course syllabus and leveraging the robust capabilities of this technology stack. Android Studio, coupled with Java, provides a comprehensive and familiar framework for Android app development, empowering our team to seamlessly translate our project goals into a tangible and functional mobile application.

GitHub

To facilitate collaborative development and version control, we employed GitHub as our primary platform. GitHub serves as the nucleus of our collaborative efforts, enabling version tracking, code reviews, and seamless integration of contributions from each team member. Leveraging the power of Git, we embrace a branching strategy that allows for concurrent development of features, minimizing conflicts and streamlining our workflow.

GitHub Desktop

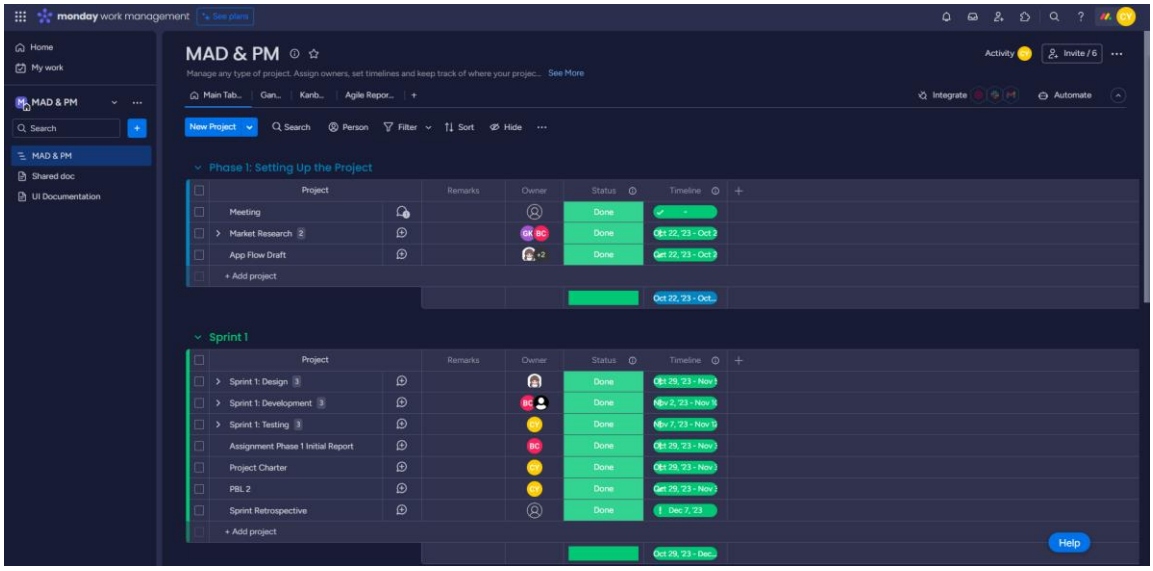
In conjunction with our use of GitHub, we employ GitHub Desktop to enhance our development workflow. GitHub Desktop provides a user-friendly interface, simplifying the management of Git repositories. This tool streamlines the process of cloning repositories, managing branches, and committing changes, making it accessible even to team members less familiar with command-line interfaces. GitHub Desktop complements our collaborative efforts by offering an intuitive way to visualize and manage the version control aspects of our project. This ensures that all team members can actively participate in version control and contribute to the development process seamlessly.

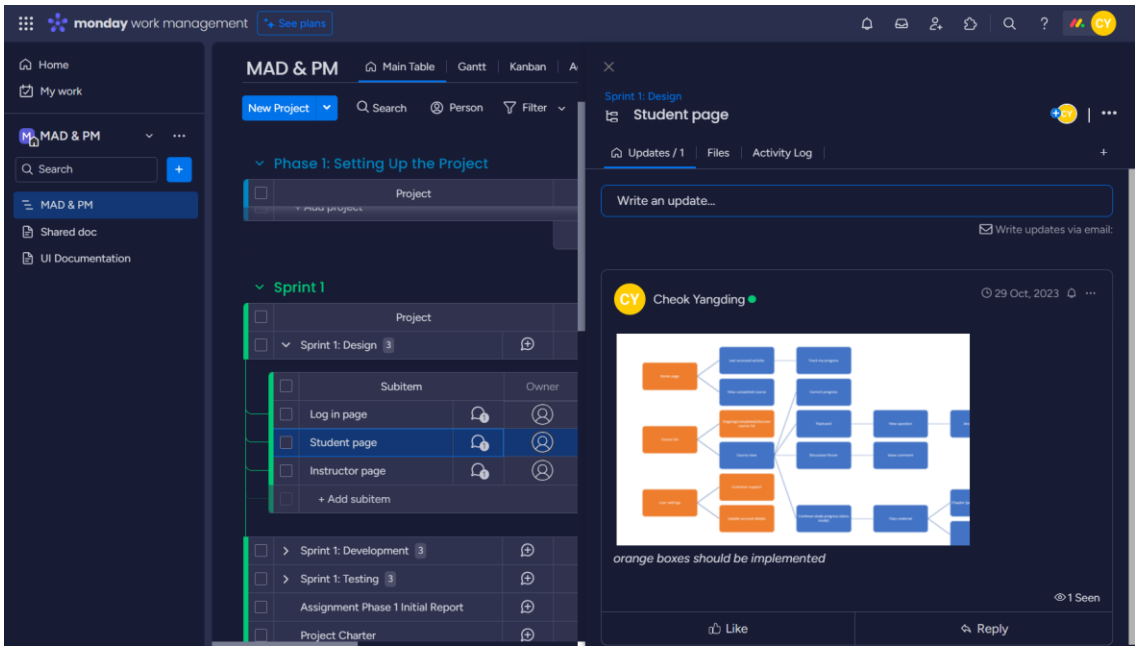
WhatsApp and Google Meet

we utilize a combination of WhatsApp and Google Meet. WhatsApp acts as an instant messaging platform for quick updates and informal discussions, ensuring real-time connectivity. Meanwhile, Google Meet serves as our virtual meeting space, accommodating regular team meetings, brainstorming sessions, sprint reviews, and retrospectives. Its video conferencing capabilities enhance our ability to discuss complex topics, share screens for code walkthroughs, and synchronize our understanding of project milestones. By combining the technical prowess of our development environment with the communicative efficiency of WhatsApp and the collaborative capabilities of Google Meet, our team maximizes its potential to deliver a successful and well-coordinated EduEmpower mobile application.

Monday.com

To streamline and manage the progress of our team, we integrated Monday.com into our toolkit. This collaborative platform provided an organized space for tracking tasks, milestones, and overall project progress. Monday.com facilitated efficient team coordination, enabling us to monitor and manage various aspects of the project in a centralized and visually accessible manner. Through the integration of Monday.com, we enhanced our project management capabilities, contributing to the successful and well-coordinated development of the EduEmpower mobile application.



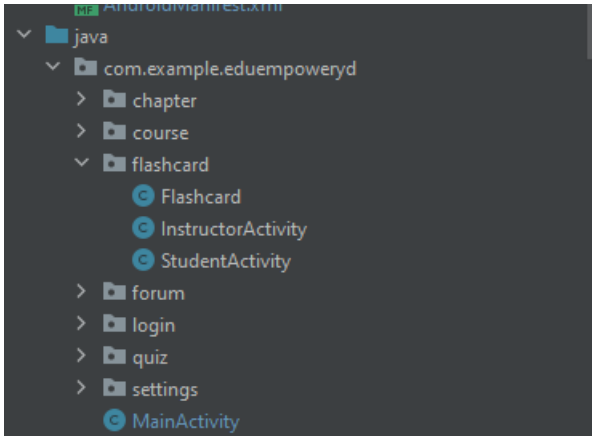


4.2 Coding Style and Convention

In the development of EduEmpower, our Android application built with Java in Android Studio, we adhere to a systematic approach for maintaining an organized and scalable codebase. Given the complexity of our project, which encompasses various modules, we adopt a hierarchical nested file structure to streamline the organization of our Java files.

4.2.1 File Structure and Organizations

Our file structure is designed to reflect the modular nature of the application, promoting clarity and ease of maintenance. A snapshot of our directory structure is outlined below:



The modular organization ensures that each module, such as 'account' or 'course,' has its dedicated space, enhancing the project's scalability and maintainability.

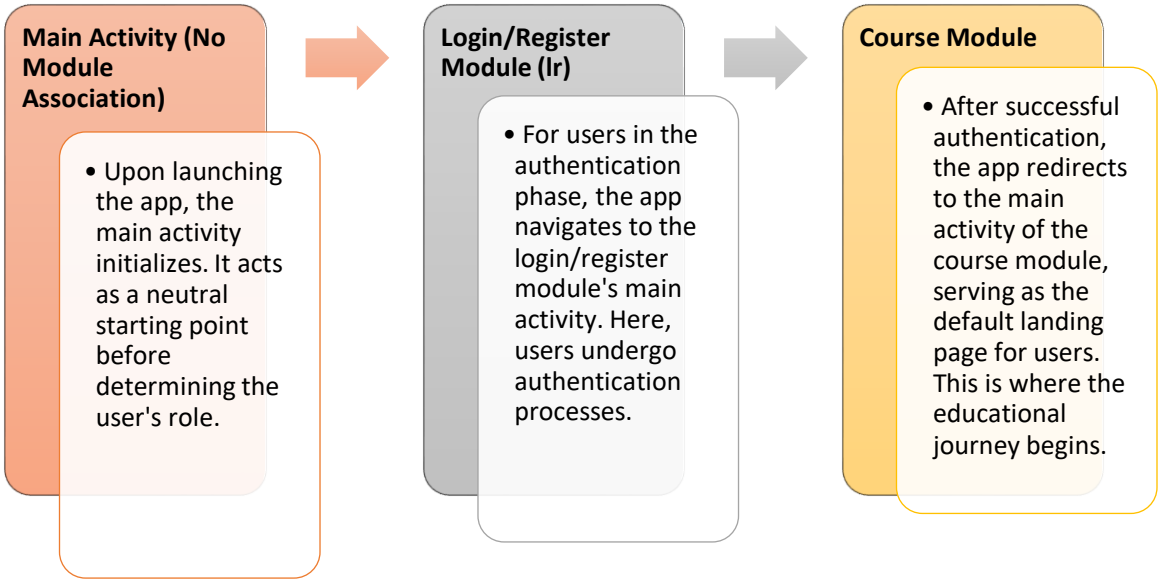
Layout Files and Naming Conventions

Due to the peculiarities of Java projects, nesting file systems within layout files proves challenging. As a workaround, we employ a systematic naming convention for our layout files. Each layout file is prefixed with the corresponding module name, followed by a brief description, all separated by underscores. For instance, 'lr_register_page.xml' signifies a layout file associated with the login/register module.

Additionally, to ensure that code is easily understandable by all team members, we've taken a meticulous approach to file naming and organization for the layout files. Layout files are specifically named to correspond with the elements in our Figma design, aligning our codebase closely with the design specifications. This practice not only aids in quick reference but also establishes a direct link between our layout files and the Figma design, promoting a harmonious integration of design and development efforts.

App Flow and Navigation

Our application's navigation follows a structured approach using activities as entry points to various modules. Two main activity files serve as gateways, catering to distinct user perspectives – one for students and another for instructors. The flow is orchestrated as follows:



Fragment-based Navigation

Within each activity, we leverage fragments to navigate between different screens efficiently. Fragments provide a modular and reusable approach to manage the app's UI components, facilitating a seamless and responsive user experience.

This strategic combination of hierarchical file structuring, naming conventions, and modular navigation contributes to the maintainability and scalability of the EduEmpower project. It ensures that the codebase remains comprehensible and adaptable as we continue to enhance the application's features and functionalities.

4.2.2 Code Comments and Documentation

Understanding the significance of collaborative coding, we emphasize the use of comments throughout our codebase. Complex algorithms and logic are accompanied by descriptive comments to explain the thought process and functionality behind each section of code. Additionally, each method is documented with a brief description of its purpose. This approach ensures that our code remains comprehensible and accessible to all team members, fostering a collaborative and inclusive development environment.

4.2.3 Code Formatting

Indentation

Consistent code formatting is crucial for readability and maintainability. To uphold these standards, we have chosen to adhere to the default formatting options provided by Android Studio. Indentation is set at 4 spaces for each level, creating a clean and standardized appearance. This uniform formatting extends to various aspects of our code, including method signatures, control structures, and variable declarations.

In addition to indentation, we maintain a consistent naming convention for variables and methods, following industry practices. This ensures that the codebase remains cohesive and comprehensible, promoting a smooth collaborative experience among team members.

Variables

We use camel case for variable names, that start with a lowercase letter and capitablize the first letter of each subsequent concatenated word. For example:

```
private List<Flashcard> flashcardsList;
private int currentCardIndex = 0;
private int easyCount = 0;
private int goodCount = 0;
private TextView questionTextView;
private TextView answerTextView;
private Button easyButton;
private Button goodButton;
private Button showAnswerButton;
private TextView congratsTextView;
private Button againButton;
private PieChart pieChart;
```

We also made sure to use descriptive names that clearly convey the purpose or content of the variable.

Methods

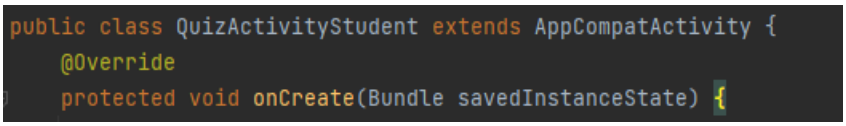
Similar to variables, camel case is used for names.

```
private void setupFlashcardView() {
    if (flashcardsList != null && !flashcardsList.isEmpty()) {
        Flashcard currentFlashcard = flashcardsList.get(currentCardIndex);
```

We also paid attention to the naming method of every method, and used verb-noun pairing method, that the name typically begin with a verb followed by a noun, and that the names reflect the action they perform.

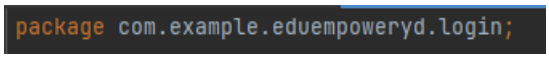
Classes

For class names, we use Pascal case, that each word starts with a capital letter.



The class names are also descriptive and indicative of the primary responsibility or role of the class within the system.

Packages



Packages names are in lowercase letters, with reverse domain naming for packages to ensure uniqueness.

4.3 Source Code & Screenshots of Key Features

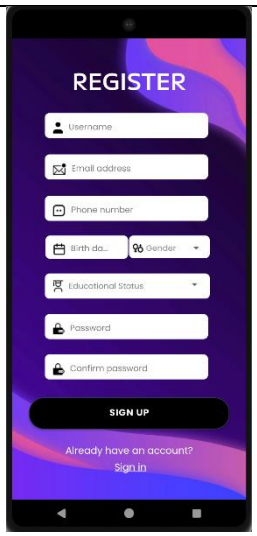
Source Code: https://github.com/yangding14/EduEmpower

4.3.1 Account Module

<table>
<tr><th>Functional Requirements</th><th>Source Code</th></tr>
<tr><td>Register account
 
</td><td>Register page

```
LrRegisterPageBinding binding = DataBindingUtil.setContentView( activity: this, R.layout.lr_register_page);
RegisterPageVM viewModel = new RegisterPageVM();
binding.setRegisterPageVM(viewModel);
binding.setLifecycleOwner(this);

TextView b1 = findViewById(R.id.txtSignIn);
b1.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent( packageContext: register_page.this, login_page.class);
                startActivity(i);
            }
        }
);
View b2 = findViewById(R.id.viewStud);
b2.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent( packageContext: register_page.this, student_register.class);
                startActivity(i);
            }
        }
);
View b3 = findViewById(R.id.viewIns);
b3.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent( packageContext: register_page.this, instructor_register.class);
                startActivity(i);
            }
        }
);
```

</td></tr>
<tr><td>Student register</td><td>Student register</td></tr>
</table>

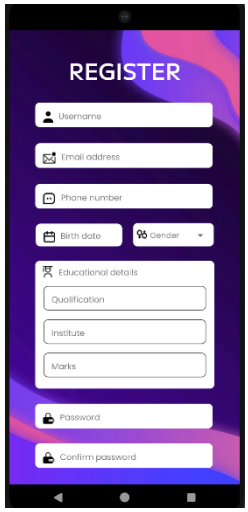


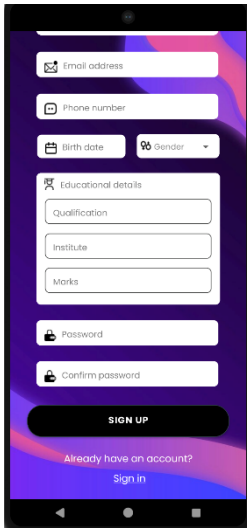

```java
SelectDate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // on below line we are getting
        // the instance of our calendar.
        final Calendar c = Calendar.getInstance();

        // on below line we are getting
        // our day, month and year.
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);

        // on below line we are creating a variable for date picker dialog.
        DatePickerDialog datePickerDialog = new DatePickerDialog(
                // on below line we are passing context.
                student_register.this,
                new DatePickerDialog.OnDateSetListener() {
                    1 usage
                    @Override
                    public void onDateSet(DatePicker view, int year,
                                          int monthOfYear, int dayOfMonth) {
                        // on below line we are setting date to our edit text.
                        SelectDate.setText(dayOfMonth + "-" + (monthOfYear + 1) + "-" + year);

                    }
                },
                // on below line we are passing year,
                // month and day for selected date in our date picker.
                year, month, day);
        // at last we are calling show to
        // display our date picker dialog.
        datePickerDialog.show();
    }
});
```

```
Spinner spinner = (Spinner) findViewById(R.id.spinnergender);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
        context: this,
        R.array.gender,
        R.layout.lr_spinner_register
);
// Specify the layout to use when the list of choices appears.
adapter.setDropDownViewResource(R.layout.lr_spinner_register);
// Apply the adapter to the spinner.
spinner.setAdapter(adapter);

spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id)
        // Change the text color for the first item
        if (position == 0) {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.GRAY);
        } else {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.BLACK);
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parentView) {
        // Do nothing here
    }
});

Spinner spinner2 = (Spinner) findViewById(R.id.spinneredu);
ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(
        context: this,
        R.array.educationalstatus,
        R.layout.lr_spinner_register
);
```

```
// Specify the layout to use when the list of choices appears.
adapter2.setDropDownViewResource(R.layout.lr_spinner_register);
// Apply the adapter to the spinner.
spinner2.setAdapter(adapter2);

spinner2.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {
        // Change the text color for the first item
        if (position == 0) {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.GRAY);
        } else {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.BLACK);
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parentView) {
        // Do nothing here
    }
});
```

Instructor register



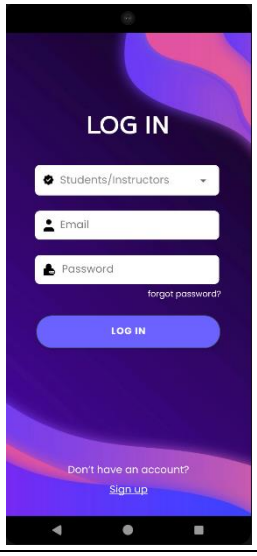



```
AppCompatButton b1 = findViewById(R.id.btnSignUp);
b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String txt_email = email.getText().toString();
                String txt_password = password.getText().toString();
                String txt_password2 = password2.getText().toString();
                String txt_username = username.getText().toString();
                String txt_phone = phone.getText().toString();
                String txt_dob = dob.getText().toString();
                String txt_gender = gender.getSelectedItem().toString();
                String txt_education = education.getSelectedItem().toString();

                if(TextUtils.isEmpty(txt_email) || TextUtils.isEmpty(txt_password)){
                    Toast.makeText(getApplicationContext(), text: "Empty Credentials", Toast.LENGTH_SHORT).show();
                } else if (txt_password.length() < 6){
                    Toast.makeText(getApplicationContext(), text: "Password too short", Toast.LENGTH_SHORT).show();
                } else if(!txt_password.equals(txt_password2)){
                    Toast.makeText(getApplicationContext(), text: "Password not match", Toast.LENGTH_SHORT).show();
                } else {
                    registerUser(txt_username, txt_email, txt_phone, txt_password, txt_dob, txt_gender, txt_education);
                }
            }
        }
);
TextView b2 = findViewById(R.id.txtSignIn);
b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i = new Intent( packageContext: student_register.this, login_page.class);
            startActivity(i);
        }
    }
);
```

```
private void registerUser(String username, String email, String phone, String password, String dob, String gender, String education
    DatabaseReference db = FirebaseDatabase.getInstance().getReference( path: "Students");
    HashMap<String, Object> map = new HashMap<>();
    map.put("Username", username);
    map.put("Email", email);
    map.put("Phone", phone);
    map.put("Password", password);
    map.put("DateOfBirth", dob);
    map.put("Gender", gender);
    map.put("Education", education);
    // Use push to generate a unique key in the Students node
    DatabaseReference newStudentRef = db.push();
    String uniqueKey = newStudentRef.getKey(); // Get the unique key
    // Use setValue on the specific reference
    newStudentRef.setValue(map)
            .addOnSuccessListener(aVoid -> {
                Log.d( tag: "StQuizInsideQuizFrag", msg: "Attempt stored successfully");
                Toast.makeText(getApplicationContext(), text: "Register successfully", Toast.LENGTH_SHORT).show();

                SharedPreferences pref = getSharedPreferences( name: "system", MODE_PRIVATE);
                SharedPreferences.Editor editor = pref.edit();
                editor.putString("uid", uniqueKey);
                editor.putString("role", "student");
                editor.apply();

                Log.d( tag: "StQuizInsideQuizFrag", msg: "dbKey: " + db.getKey());

                startActivity(new Intent( packageContext: student_register.this, student_survey.class));
            })
            .addOnFailureListener(e -> {
                // Show error message
                Toast.makeText(getApplicationContext(), text: "Error: " + e.getMessage(), Toast.LENGTH_LONG).show();
            });
}
```

Instructor register page

Login account




```
SelectDate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(
                context: instructor_register.this, new DatePickerDialog.OnDateSetListener() {
                    1 usage
                    @Override
                    public void onDateSet(DatePicker view, int year,
                                          int monthOfYear, int dayOfMonth) {
                        SelectDate.setText(dayOfMonth + "-" + (monthOfYear + 1) + "-" + year);
                    }
                }, year, month, day);
        datePickerDialog.show();
    }
});

Spinner spinner = (Spinner) findViewById(R.id.spinnergender);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
         context: this,
        R.array.gender,
        R.layout.lr_spinner_register
);
// Specify the layout to use when the list of choices appears.
adapter.setDropDownViewResource(R.layout.lr_spinner_register);
// Apply the adapter to the spinner.
spinner.setAdapter(adapter);
```

```
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {
        // Change the text color for the first item
        if (position == 0) {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.GRAY);
        } else {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.BLACK);
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parentView) {
        // Do nothing here
    }
});
```

```
AppCompatButton b1 = findViewById(R.id.btnSignUp);
b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String txt_email = email.getText().toString();
                String txt_phone = phone.getText().toString();
                String txt_username = username.getText().toString();
                String txt_password = password.getText().toString();
                String txt_dob = dob.getText().toString();
                String txt_gender = gender.getSelectedItem().toString();
                String txt_qualification = qualification.getText().toString();
                String txt_institute = institute.getText().toString();
                String txt_marks = marks.getText().toString();

                if(TextUtils.isEmpty(txt_email) || TextUtils.isEmpty(txt_password)){
                    Toast.makeText(getApplicationContext(), text: "Empty Credentials", Toast.LENGTH_SHORT).show();
                } else if (txt_password.length() < 6){
                    Toast.makeText(getApplicationContext(), text: "Password too short", Toast.LENGTH_SHORT).show()
                } else{
                    registerUser(txt_username, txt_email, txt_phone, txt_password, txt_dob, txt_gender,
                            txt_qualification, txt_institute, txt_marks);
                }

            }
        }
);
```

```
private void registerUser(String txtUsername, String txtEmail, String txtPhone, String txtPassword,
                          String txtDob, String txtGender, String txtQualification, String txtInstitute, String txtMarks) {
    DatabaseReference db = FirebaseDatabase.getInstance().getReference( path: "Instructors");

    HashMap <String, Object> map = new HashMap<>();
    map.put("Username", txtUsername);
    map.put("Email", txtEmail);
    map.put("Phone", txtPhone);
    map.put("Password", txtPassword);
    map.put("DateOfBirth", txtDob);
    map.put("Gender", txtGender);
    map.put("Qualification", txtQualification);
    map.put("Institute", txtInstitute);
    map.put("Marks", txtMarks);

    // Use push to generate a unique key in the Students node
    DatabaseReference newInstructorRef = db.push();
    String uniqueKey = newInstructorRef.getKey(); // Get the unique key
```

```
    // Use setValue on the specific reference
    newInstructorRef.setValue(map)
            .addOnSuccessListener(aVoid -> {
                Log.d( tag: "StQuizInsideQuizFrag",  msg: "Attempt stored successfully");
                Toast.makeText(getApplicationContext(), text: "Register successfully", Toast.LENGTH_SHORT).show();

                SharedPreferences pref = getSharedPreferences( name: "system", MODE_PRIVATE);
                SharedPreferences.Editor editor = pref.edit();
                editor.putString("uid", uniqueKey);
                editor.putString("role", "instructor");
                editor.apply();

                Log.d( tag: "StQuizInsideQuizFrag",  msg: "dbKey: " + db.getKey());

                startActivity(new Intent( packageContext: instructor_register.this,
                        com.example.eduempoweryd.course.ListofCoursesActivity.class));
            })
            .addOnFailureListener(e -> {
                // Show error message
                Toast.makeText(getApplicationContext(), text: "Error: " + e.getMessage(), Toast.LENGTH_LONG).show()
            });
}

public void onClick(View v){
    Intent i = new Intent( packageContext: instructor_register.this, login_page.class);
    startActivity(i);
}
```

Login Page

```java
LrLoginPageBinding binding = DataBindingUtil.setContentView( activity: this, R.layout.lr_login_page);
LoginPageVM viewModel = new LoginPageVM();
binding.setLoginPageVM(viewModel);
binding.setLifecycleOwner(this);
Spinner spinner = (Spinner) findViewById(R.id.spinnerstud);
// Create an ArrayAdapter using the string array and a default spinner layout.
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
        context: this,
        R.array.user_types,
        R.layout.lr_spinner_list
);
// Specify the layout to use when the list of choices appears.
adapter.setDropDownViewResource(R.layout.lr_spinner_list);
// Apply the adapter to the spinner.
spinner.setAdapter(adapter);

spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {
        // Change the text color for the first item
        if (position == 0) {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.GRAY);
        } else {
            ((TextView) parentView.getChildAt( index: 0)).setTextColor(Color.BLACK);
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parentView) {
        // Do nothing here
    }
});
```

```java
TextView b1 = findViewById(R.id.txtForgotpassword);
b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // inflate the layout of the popup window
                LayoutInflater inflater = (LayoutInflater)
                        getSystemService(LAYOUT_INFLATER_SERVICE);
                View popupView = inflater.inflate(R.layout.lr_forgot_password,  root: null);

                // create the popup window
                int width = LinearLayout.LayoutParams.WRAP_CONTENT;
                int height = LinearLayout.LayoutParams.WRAP_CONTENT;
                boolean focusable = true; // lets taps outside the popup also dismiss it
                final PopupWindow popupWindow = new PopupWindow(popupView, width, height, focusable);

                // show the popup window
                // which view you pass in doesn't matter, it is only used for the window tolken
                popupWindow.setAnimationStyle(R.style.popup_window_animation);
                popupWindow.showAtLocation(v, Gravity.CENTER,  x: 0,  y: 0);

                TextView txtSend = popupView.findViewById(R.id.txtSend);
                txtSend.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) { popupWindow.dismiss(); }
                });
            }
        }
);
```

```
AppCompatButton b2 = findViewById(R.id.btnLogInOne);
b2.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String txt_email = email.getText().toString();
                String txt_role = spinner.getSelectedItem().toString();
                String txt_password = password.getText().toString();
                Log.d( tag: "Email", txt_email);
                Log.d( tag: "Password", txt_password);
                Log.d( tag: "Role", txt_role);
                if(txt_role.equals("Students")){
                    if(!txt_email.isEmpty()){
                        studentLogin(txt_email, txt_password);
                    }
                    else{
                        Toast.makeText(getApplicationContext(),  text: "Please enter username", Toast.LENGTH_SHORT).show();
                    }
                } else if (txt_role.equals("Instructors")){
                    if(!txt_email.isEmpty()){
                        instructorLogin(txt_email, txt_password);
                    }
                    else{
                        Toast.makeText(getApplicationContext(),  text: "Please enter username", Toast.LENGTH_SHORT).show();
                    }
                } else{
                    Toast.makeText(getApplicationContext(),  text: "Please enter your role", Toast.LENGTH_SHORT).show();
                }
            }
        }
);
```

```
TextView b3 = findViewById(R.id.txtSignUp);
b3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent( packageContext: login_page.this, register_page.class);
                startActivity(i);
            }
        }
);
```

```
private void instructorLogin(String email,String password) {
    // get password from database
    DatabaseReference db = FirebaseDatabase.getInstance().getReference( path: "Instructors");
    db.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            for(DataSnapshot child: snapshot.getChildren()){
                if(child.child( path: "Email").getValue().toString().equals(email)){
                    String pass = child.child( path: "Password").getValue().toString();
                    if(pass.equals(password)){
                        SharedPreferences pref = getSharedPreferences( name: "system", MODE_PRIVATE);
                        SharedPreferences.Editor editor = pref.edit();
                        editor.putString("uid", child.getKey());
                        editor.putString("role", "instructor");
                        editor.apply();

                        startActivity(new Intent( packageContext: login_page.this, com.example.eduempoweryd.course.ListofCoursesActivity.class))
                        finish();
                    }
                }
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            // calling on cancelled method when we receive any error or we are not able to get the data.
            Log.e( tag: "StQuizQuestionsList",  msg: "Fail to get data.");
        }
    });
}
```

```
private void studentLogin(String email, String password) {
    // get password from database
    DatabaseReference db = FirebaseDatabase.getInstance().getReference( path: "Students");
    db.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            for(DataSnapshot child: snapshot.getChildren()){
                if(child.child( path: "Email").getValue().toString().equals(email)){
                    String pass = child.child( path: "Password").getValue().toString();
                    if(pass.equals(password)){
                        SharedPreferences pref = getSharedPreferences( name: "system", MODE_PRIVATE);
                        SharedPreferences.Editor editor = pref.edit();
                        editor.putString("uid", child.getKey());
                        editor.putString("role", "student");
                        editor.apply();

                        startActivity(new Intent( packageContext: login_page.this, com.example.eduempoweryd.course.MainActivity.class))
                        finish();
                    }

                }
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            // calling on cancelled method when we receive any error or we are not able to get the data.
            Log.e( tag: "StQuizQuestionsList",  msg: "Fail to get data.");
        }
    });
}
```

User settings:

Students' settings

Account Fragment:

To retrieve data from database to update the name and email

```
// Show profile username and email
SharedPreferences preferences = getActivity().getSharedPreferences( name: "system", MODE_PRIVATE)
String uid = preferences.getString( key: "uid",  defValue: "null");

DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Students");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String username = String.valueOf(snapshot.child( path: "Username").getValue());
            String email = String.valueOf(snapshot.child( path: "Email").getValue());
            userName.setText(username);
            userEmail.setText(email);
        }
    }
});
```

To enter each update activity, FAQ and lot out

```
//Button Customer Support
cs_button.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        ((MainActivity)getActivity()).enterCustomerSupport(view);
    }
});

// Button Change Username
ImageButton btnChangeName = view.findViewById(R.id.btn_acc_username);

btnChangeName.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        Intent i = new Intent(getActivity(), St_UpdateUsernameActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText().toString());
        startActivity(i);
    }
});

// Button Change Email
ImageButton btnChangeEmail = view.findViewById(R.id.btn_acc_email);
btnChangeEmail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getActivity(), St_UpdateEmailActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText().toString());
        startActivity(i);
    }
});
```

Student Update username	

```
// Button Upload DOB
ImageButton btnChangeDOB = view.findViewById(R.id.btn_acc_dob);
btnChangeDOB.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getActivity(), St_UpdateDOBActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText().toString());
        startActivity(i);
    }
});

// Button Change Password
ImageButton btnChangePassword = view.findViewById(R.id.btn_acc_change_password);
btnChangePassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getActivity(), St_UpdatePasswordActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText().toString());
        startActivity(i);
    }
});

// Change Phone
ImageButton btnChangePhoneNumber = view.findViewById(R.id.btn_acc_phone);
btnChangePhoneNumber.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getActivity(), St_UpdatePhoneActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText().toString());
        startActivity(i);
    }
});
```


```
// Upload Gender
ImageButton btnChangeGender = view.findViewById(R.id.btn_acc_gender);
btnChangeGender.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getActivity(), St_UpdateGenderActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText().toString());
        startActivity(i);
    }
});

// Enter Customer SUpport
cs_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(getActivity(), CustomerSupportActivity.class);
        startActivity(i);
    }
});

// Log out
btn_logout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(getActivity(), login_page.class));
        getActivity().finish();
    }
});
```

Update username activity

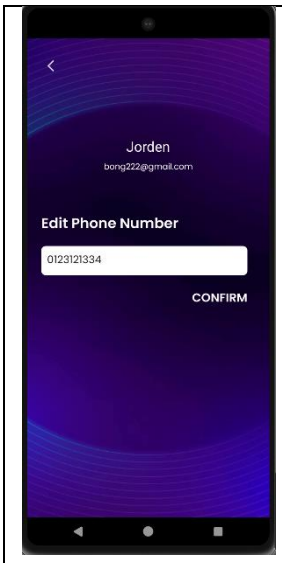
Student update email address

Student update phone number

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Update user profile
updateProfile(userName, userEmail);

//Retrieve data for ET
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Students");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String username = String.valueOf(snapshot.child( path: "Username").getValue());
            editUsername.setText(username);
        }
    }
});

backButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { finish(); }
});
```

```
    confirmButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String txt_username = editUsername.getText().toString();
            HashMap map = new HashMap();
            map.put("Username", txt_username);

            if(!txt_username.isEmpty()){
                FirebaseDatabase.getInstance().getReference( path: "Students")
                        .child(uid).updateChildren(map);
                  Intent i = new Intent(St_UpdateEmailActivity.this, MainActivity.class);
                  i.putExtra("Navigation", "account_fragment");
                  startActivity(i);
                finish();
            } else{
                finish();
            }
        }
    });
}

1 usage
private void updateProfile(TextView userName, TextView userEmail) {
    String name = getIntent().getStringExtra( name: "Username");
    String email = getIntent().getStringExtra( name: "Email");
    userName.setText(name);
    userEmail.setText(email);
}
```

Update email address activity

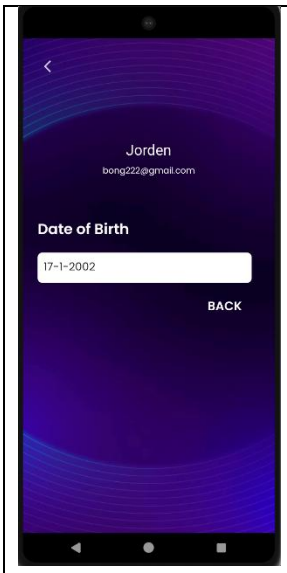


Student show date of birth

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Update user profile
updateProfile(userName, userEmail);

//Retrieve data for ET
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Students");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String email = String.valueOf(snapshot.child( path: "Email").getValue());
            editEmail.setText(email);
        }
    }
});

backButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { finish(); }
});
```

```
    confirmButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String txt_email = editEmail.getText().toString();
            HashMap map = new HashMap();
            map.put("Email", txt_email);

            if(!txt_email.isEmpty()){
                FirebaseDatabase.getInstance().getReference( path: "Students")
                        .child(uid).updateChildren(map);
                  Intent i = new Intent(St_UpdateEmailActivity.this, MainActivity.class);
                  i.putExtra("Navigation", "account_fragment");
                  startActivity(i);
                finish();
            } else{
                finish();
            }
        }
    });
}

1 usage
private void updateProfile(TextView userName, TextView userEmail) {
    String name = getIntent().getStringExtra( name: "Username");
    String email = getIntent().getStringExtra( name: "Email");
    userName.setText(name);
    userEmail.setText(email);
}
```

Update phone number activity




```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Update user profile
updateProfile(userName, userEmail);

//Retrieve data for ET
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Students");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String phone = String.valueOf(snapshot.child( path: "Phone").getValue());
            editPhone.setText(phone);
        }
    }
});

backButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { finish(); }
});
```

Student show gender


```
    confirmButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String txt_phone = editPhone.getText().toString();
            HashMap map = new HashMap();
            map.put("Phone", txt_phone);

            if(!txt_phone.isEmpty()){
                FirebaseDatabase.getInstance().getReference( path: "Students")
                        .child(uid).updateChildren(map);
                  Intent i = new Intent(St_UpdateEmailActivity.this, MainActivity.class);
                  i.putExtra("Navigation", "account_fragment");
                  startActivity(i);
                finish();
            } else{
                finish();
            }
        }
    });
}

1 usage
private void updateProfile(TextView userName, TextView userEmail) {
    String name = getIntent().getStringExtra( name: "Username");
    String email = getIntent().getStringExtra( name: "Email");
    userName.setText(name);
    userEmail.setText(email);
}
```

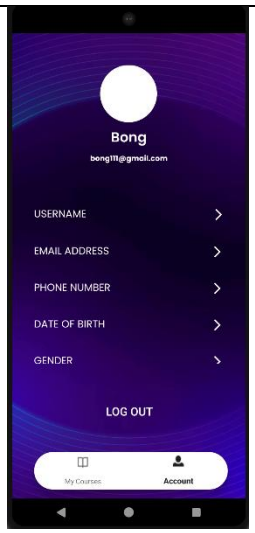
Show date of birth activity

Student update password


```
    SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
    String uid = preferences.getString( key: "uid",  defValue: "");

    updateProfile(userName, userEmail);

    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Students");
    reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DataSnapshot> task) {
            if (task.isSuccessful()) {
                DataSnapshot snapshot = task.getResult();
                String dob = String.valueOf(snapshot.child( path: "DateOfBirth").getValue());
                editDOB.setText(dob);
            }
        }
    });
    backButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { finish(); }
    });
    returnButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { finish(); }
    });
}

1 usage
private void updateProfile(TextView userName, TextView userEmail) {
    String name = getIntent().getStringExtra( name: "Username");
    String email = getIntent().getStringExtra( name: "Email");
    userName.setText(name);
    userEmail.setText(email);
}
```

Show gender activity

Instructors User Settings



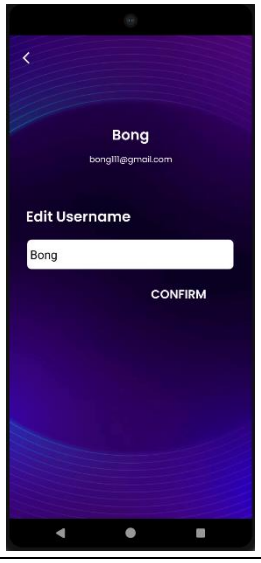

```
    SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
    String uid = preferences.getString( key: "uid",  defValue: "");

    updateProfile(userName, userEmail);

    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Students");
    reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DataSnapshot> task) {
            if (task.isSuccessful()) {
                DataSnapshot snapshot = task.getResult();
                String gender = String.valueOf(snapshot.child( path: "Gender").getValue());
                editGender.setText(gender);
            }
        }
    });
    backButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { finish(); }
    });
    returnButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { finish(); }
    });
}

1 usage
private void updateProfile(TextView userName, TextView userEmail) {
    String name = getIntent().getStringExtra( name: "Username");
    String email = getIntent().getStringExtra( name: "Email");
    userName.setText(name);
    userEmail.setText(email);
}
```

Change Password activity

```
    SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
    String uid = preferences.getString( key: "uid",  defValue: "");

    updateProfile(userName, userEmail);

    backButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { finish(); }
    });
    confirmButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String txt_password = editPassword.getText().toString();
            HashMap map = new HashMap();
            map.put("Password", txt_password);

            if(!txt_password.isEmpty()){
                FirebaseDatabase.getInstance().getReference( path: "Students")
                        .child(uid).updateChildren(map);
                finish();
            } else{
                finish();
            }
        }
    });
}

1 usage
private void updateProfile(TextView userName, TextView userEmail) {
    String name = getIntent().getStringExtra( name: "Username");
    String email = getIntent().getStringExtra( name: "Email");
    userName.setText(name);
    userEmail.setText(email);
}
```

Instructor Account Settings

Instructor update username



Instructor update email address


```
bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
    1 usage
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {

        if (item.getItemId() == R.id.Course) {
            startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.course.InCourseList.class));
            return true;
        } else if (item.getItemId() == R.id.Account) {
            return true;
        } else
            return false;
    }

});

// Show profile username and email
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String username = String.valueOf(snapshot.child( path: "Username").getValue());
            String email = String.valueOf(snapshot.child( path: "Email").getValue());
            userName.setText(username);
            userEmail.setText(email);
        }
    }
});
```

```
updatePhone.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdatePhoneActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText()).toString();
        startActivity(i);
    }
});

updateEmail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdateEmailActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText()).toString();
        startActivity(i);
    }
});

updatePassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdatePasswordActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText()).toString();
        startActivity(i);
    }
});
```

Instructor update phone number


```
updateGender.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdateGenderActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText()).toString();
        startActivity(i);
    }
});

updateEducation.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdateEducationActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText()).toString();
        startActivity(i);
    }
});

updateName.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdateUsernameActivity.class);
        i.putExtra( name: "Username", userName.getText().toString());
        i.putExtra( name: "Email", userEmail.getText()).toString();
        startActivity(i);
    }
});
```

```
    updateDOB.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i = new Intent( packageContext: UserSettingsActivity.this, UpdateDOBActivity.class);
            i.putExtra( name: "Username", userName.getText().toString());
            i.putExtra( name: "Email", userEmail.getText()).toString();
            startActivity(i);
        }
    });

    buttonLogout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
            SharedPreferences.Editor editor = preferences.edit();
            editor.clear();
            editor.apply();
            Intent i = new Intent( packageContext: UserSettingsActivity.this, login_page.class);
            startActivity(i);
        }
    });
}

no usages
public void open_InstructorCourseList(){
    Intent intent = new Intent( packageContext: this , com.example.eduempoweryd.course.InCourseList.class)
    startActivity(intent);
}
```

Update username

Instructor show date of birth


```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Initialise
userName = findViewById(R.id.userName2);
userEmail = findViewById(R.id.userEmail2);
buttonConfirm = findViewById(R.id.btnConfirmEmail);
editUsername = findViewById(R.id.EditUsername);

//Show profile data
String name = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userEmail.setText(email);
userName.setText(name);

//Show username
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String name = String.valueOf(snapshot.child( path: "Username").getValue());
            editUsername.setText(name);
        }
    }
});
```

Instructor show gender


```
//Handle button event
buttonConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String txt_username = editUsername.getText().toString();
        HashMap map = new HashMap();
        map.put("Username", txt_username);

        if(!txt_username.isEmpty()){
            FirebaseDatabase.getInstance().getReference( path: "Instructors")
                    .child(uid).updateChildren(map);
            startActivity(new Intent( packageContext: UpdateUsernameActivity.this, UserSettingsActivity.class));
            finish();
        } else{
            finish();
        }
    }
});

ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class))
    }
});
```

Update email address

Instructor update education details


```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Initialise
userName = findViewById(R.id.userName2);
userEmail = findViewById(R.id.userEmail2);
buttonConfirm = findViewById(R.id.btnConfirmEmail);
editEmail = findViewById(R.id.EditUsername);

//Show profile data
String name = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userEmail.setText(email);
userName.setText(name);

//Show email
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String email = String.valueOf(snapshot.child( path: "Email").getValue());
            editEmail.setText(email);
        }
    }
});
```

```
//Handle button event
buttonConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String txt_email = editEmail.getText().toString();
        HashMap map = new HashMap();
        map.put("Email", txt_email);

        if(!txt_email.isEmpty()){
            FirebaseDatabase.getInstance().getReference( path: "Instructors")
                    .child(uid).updateChildren(map);
            startActivity(new Intent( packageContext: UpdateEmailActivity.this, UserSettingsActivity.class));
            finish();
        } else{
            finish();
        }
    }
});

ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class))
    }
});
```

Update phone number

Instructor update password


```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Initialise
userName = findViewById(R.id.userName5);
userEmail = findViewById(R.id.userEmail5);
buttonConfirm = findViewById(R.id.btnConfirmPhone);
editPhone = findViewById(R.id.EditPhone);

//Show profile data
String name = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userName.setText(name);
userEmail.setText(email);

// Get data (Phone) from database
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String phone = String.valueOf(snapshot.child( path: "Phone").getValue());
            editPhone.setText(phone);
        }
    }
});
```

```
//Button handle event
buttonConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String txt_phone = editPhone.getText().toString();
        HashMap map = new HashMap();
        map.put("Phone", txt_phone);

        if(!txt_phone.isEmpty()){
            FirebaseDatabase.getInstance().getReference( path: "Instructors")
                    .child(uid).updateChildren(map);
            startActivity(new Intent( packageContext: UpdatePhoneActivity.this, UserSettingsActivity.class));
            finish();
        }else{
            finish();
        }
    }
});

ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class))
    }
});
```

Show date of birth

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

userEmail = findViewById(R.id.userEmail4);
userName = findViewById(R.id.userName4);
editDOB = findViewById(R.id.EditDOB);


// Set profile data: name and email
String username = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userName.setText(username);
userEmail.setText(email);

// Show DOB from database
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String dob = String.valueOf(snapshot.child( path: "DateOfBirth").getValue());
            editDOB.setText(dob);
        }
    }
});


ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class));
    }
});
```

Show gender

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

userName = findViewById(R.id.userName3);
userEmail = findViewById(R.id.userEmail3);
editGender = findViewById(R.id.EditGender);

// Show profile data
String name = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userEmail.setText(email);
userName.setText(name);

// Get data (Gender) from database
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String gender = String.valueOf(snapshot.child( path: "Gender").getValue());
            editGender.setText(gender);
        }
    }
});

ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class))
    }
});
```

Update education details

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");

//Initialise
userName = findViewById(R.id.userName8);
userEmail = findViewById(R.id.userEmail8);
buttonConfirm = findViewById(R.id.btnConfirmEducation);
editQualification = findViewById(R.id.EditQualification);
editInstitute = findViewById(R.id.EditInstitute);
editMarks = findViewById(R.id.EditMarks);

//Show profile data
String name = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userName.setText(name);
userEmail.setText(email);

//Show education details
DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Instructors");
reference.child(uid).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DataSnapshot> task) {
        if (task.isSuccessful()) {
            DataSnapshot snapshot = task.getResult();
            String qualification = String.valueOf(snapshot.child( path: "Qualification").getValue());
            String institute = String.valueOf(snapshot.child( path: "Institute").getValue());
            String marks = String.valueOf(snapshot.child( path: "Marks").getValue());
            editQualification.setText(qualification);
            editInstitute.setText(institute);
            editMarks.setText(marks);
        }
    }
});
```

```
//Button handle event
buttonConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String txt_qualification = editQualification.getText().toString();
        String txt_institute = editInstitute.getText().toString();
        String txt_marks = editMarks.getText().toString();

        HashMap map = new HashMap();
        map.put("Qualification", txt_qualification);
        map.put("Institute", txt_institute);
        map.put("Marks", txt_marks);

        if(!txt_qualification.isEmpty() || !txt_institute.isEmpty() || !txt_marks.isEmpty()){
            FirebaseDatabase.getInstance().getReference( path: "Instructors")
                    .child(uid).updateChildren(map);
            startActivity(new Intent( packageContext: UpdateEducationActivity.this, UserSettingsActivity.class));
            finish();
        }else{
            finish();
        }
    }
});

ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class));
    }
});
```

Update password

```
SharedPreferences preferences = getSharedPreferences( name: "system", MODE_PRIVATE);
String uid = preferences.getString( key: "uid",  defValue: "");
//Initialise
userName = findViewById(R.id.userName7);
userEmail = findViewById(R.id.userEmail7);
buttonConfirm = findViewById(R.id.btnConfirmPassword);
editPassword = findViewById(R.id.EditPassword2);

//Show profile data
String name = getIntent().getStringExtra( name: "Username");
String email = getIntent().getStringExtra( name: "Email");
userName.setText(name);
userEmail.setText(email);

//Button handle event
buttonConfirm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String txt_password = editPassword.getText().toString();
        HashMap map = new HashMap();
        map.put("Password", txt_password);

        if(!txt_password.isEmpty()){
            FirebaseDatabase.getInstance().getReference( path: "Instructors")
                    .child(uid).updateChildren(map);
            startActivity(new Intent( packageContext: UpdatePasswordActivity.this, UserSettingsActivity.class))
            finish();
        }else{
            finish();
        }
    }
});
```

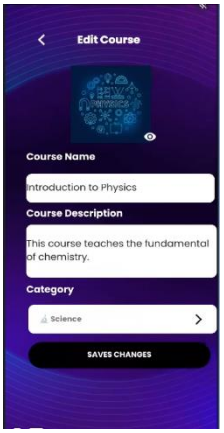
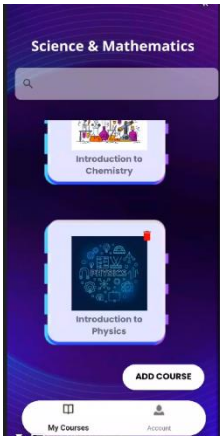
```
ImageButton backbutton = findViewById(R.id.btn_cs_back);
backbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getApplicationContext(), com.example.eduempoweryd.settings.instructor.UserSettingsActivity.class));
    }
});
```

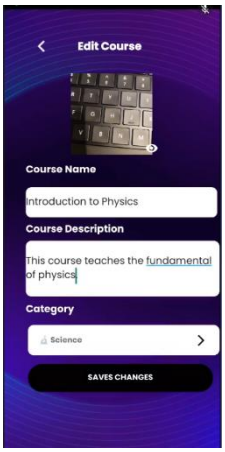
Customer Service / FAQ

Customer support / FAQ

```
3 usages
private CourseActivityCustomerSupportBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = CourseActivityCustomerSupportBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    binding.btnCsBack.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { finish(); }
    });
}
```

4.3.2 Course Module

Functional Requirements	Source Code
Manage Course    	**Instructor can add course** `private void renderListOfCourses(){` `DatabaseReference db = FirebaseDatabase.getInstance().getReference( path: "Course");` `yangding14 +1 *` `db.addListenerForSingleValueEvent(new ValueEventListener() {` `no usages` `private String courseId;` `no usages` `private String courseTitle;` `no usages` `private String courseDesc;` `no usages` `private String category;` `no usages` `private String uri;` `yangding14 +1 *` `@Override` `public void onDataChange(@NonNull DataSnapshot snapshot) {` `for (DataSnapshot dataSnapshot : snapshot.getChildren()) {` `courses.add(new Course(dataSnapshot.getKey(), dataSnapshot.child( path: "courseTitle").getValue(String.class),` `dataSnapshot.child( path: "courseDesc").getValue(String.class),` `dataSnapshot.child( path: "category").getValue(String.class),` `dataSnapshot.child( path: "uri").getValue(String.class)));` `Log.d( tag: "StQuizQuestionsList", msg: "Success to get data.");` `Log.d( tag: "StQuizQuestionsList", msg: "Value is: " + dataSnapshot.getValue());` `}` `Log.d( tag: "StQuizQuestionsList", msg: "courses size: " + courses.size());` `for (Course course : courses) {` `View courseView = getLayoutInflater().inflate(R.layout.course_one_course, coursesContainer, attachToRoot: false);` `ImageView imageView = courseView.findViewById(R.id.image);` `// Use Glide to load the image from the URI` `Glide.with(getApplicationContext()) RequestManager` `.load(course.getUri()) RequestBuilder<Drawable>` `.into(imageView);` `TextView courseTitleTextView = courseView.findViewById(R.id.courseTitle);` `courseTitleTextView.setText(course.getCourseTitle());`




```
            // Set OnClickListener for each courseView
            yangding14 +1
            courseView.setOnClickListener(new View.OnClickListener() {
                yangding14 +1
                @Override
                public void onClick(View v) {
                    // Handle the click event for the courseView
                    // For example, you can open a new activity with details of the selected course
                    SharedPreferences sharedPreferences = getSharedPreferences( name: "system", MODE_PRIVATE);
                    sharedPreferences.edit().putString( s: "courseId", course.getCourseId()).apply();

                    Intent intent = new Intent( packageContext: ListofCoursesActivity.this, InCourseViewActivity.class);
                    intent.putExtra( name: "courseId", course.getCourseId()); // Pass any relevant data
                    startActivity(intent);
                }
            });

            coursesContainer.addView(courseView);
        }
    }

    yangding14
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        // calling on cancelled method when we receive any error or we are not able to get the data.
        Log.e( tag: "StQuizQuestionsList", msg: "Fail to get data.");
    }
});
```

```
1 usage   yangding14 +1
public void addNewCourse(View view) {
    SharedPreferences sharedPreferences = getSharedPreferences( name: "system", MODE_PRIVATE);
    sharedPreferences.edit().putString( s: "courseId",  s1: null).apply();

    startActivity(new Intent(getApplicationContext(), InEditCourseActivity.class));
```

Instructor can edit course like edit image and text.

Resize the image to a square aspect ratio before uploading

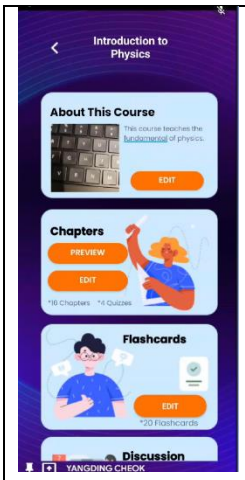
```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_CODE) {
        if (resultCode == RESULT_OK && data != null) {
            // Retrieve the updated text directly from the Intent data
            String selectedText = data.getStringExtra( name: "selectedText");
            if (selectedText != null && !selectedText.isEmpty()) {
                textView.setText(selectedText);
            }
        }
    }

    // code related to upload image
    if (requestCode == 2 && resultCode == RESULT_OK && data != null) {
        imageUri = data.getData();
        // Resize the image to a square aspect ratio before uploading
        try {
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(this.getContentResolver(), imageUri);
            int width = bitmap.getWidth();
            int height = bitmap.getHeight();
            int size = Math.min(width, height);
            Bitmap squareBitmap = Bitmap.createBitmap(bitmap,  x: (width - size) / 2,  y: (height - size) / 2, size, size);
```

```
            // Convert the Bitmap to Uri
            imageUri = getImageUri( context: this, squareBitmap);

            ImageView imageView = findViewById(R.id.imageView);
            imageView.setImageURI(imageUri);

            // Proceed with uploading the resized image to Firebase Storage
            if (imageUri != null) {
                uploadToFirebase(imageUri);
            } else {
                Toast.makeText( context: InEditCourseActivity.this,  text: "Failed to get image URI", Toast.LENGTH_SHORT).show();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

push to firebase

```
private void uploadToFirebase(Uri uri) {
    final StorageReference fileRef = reference.child( pathString: System.currentTimeMillis() + "." + getFileExtension(uri));
    yangding14
    fileRef.putFile(uri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        yangding14
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            yangding14
            fileRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                yangding14
                @Override
                public void onSuccess(Uri uri) {
                    Model model = new Model(uri.toString());
                    Log.d( tag: "tag",  msg: "onSuccess: Uploaded Image URl is " + uri.toString());
                    uriToStore = uri.toString();
                    progressBar.setVisibility(View.INVISIBLE);
                    Toast.makeText( context: InEditCourseActivity.this,  text: "Uploaded Successfully", Toast.LENGTH_SHORT).show();
                }
            });
        }
    }).addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
        yangding14
        @Override
        public void onProgress(@NonNull UploadTask.TaskSnapshot snapshot) {
            progressBar.setVisibility(View.VISIBLE);
        }
    yangding14
    }).addOnFailureListener(new OnFailureListener() {
        yangding14
        @Override
        public void onFailure(@NonNull Exception e) {
            progressBar.setVisibility(View.INVISIBLE);
            Toast.makeText( context: InEditCourseActivity.this,  text: "Uploading Failed !!", Toast.LENGTH_SHORT).show();
        }
    });
}

1 usage  yangding14
private String getFileExtension(Uri mUri) {
    ContentResolver cr = getContentResolver();
    MimeTypeMap mime = MimeTypeMap.getSingleton();
    return mime.getExtensionFromMimeType(cr.getType(mUri));

}
```

```
//upload text into firebase
1 usage  yangding14
public void uploadData() {
    String title = edit1.getText().toString();
    String desc = edit2.getText().toString();
    String selectedText = textView.getText().toString();

    SharedPreferences sharedPreferences = getSharedPreferences( name: "system", MODE_PRIVATE);
    String courseid = sharedPreferences.getString( s: "courseId",  s1: "");

    DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference( path: "Course");
    if(courseid.isEmpty() || courseid.equals(null)){
        DatabaseReference newRef = FirebaseDatabase.getInstance().getReference( path: "Course").push();
        courseid = newRef.getKey(); // Get the generated key
    }

    // Create a Map to hold all the data together
    Map<String, Object> courseData = new HashMap<>();
    courseData.put( k: "courseTitle", title);
    courseData.put( k: "courseDesc", desc);
    courseData.put( k: "category", selectedText);
    courseData.put( k: "uri", uriToStore);
```

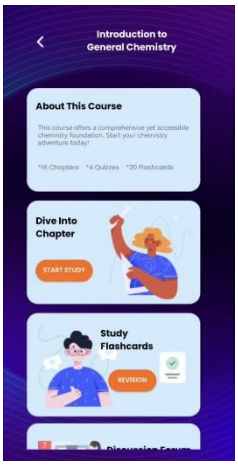



```
    // Save all the data under the generated unique key
    databaseRef.child(courseid).setValue(courseData)
            yangding14
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                yangding14
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText( context: InEditCourseActivity.this,  text: "Course details uploaded to Firebase", Toast.LENGTH_SHORT).sho
                    } else {
                        Toast.makeText( context: InEditCourseActivity.this,  text: "Failed to upload course details to Firebase", Toast.LENGTH_SH
                    }
                }
            });
}
```

```
private Uri getImageUri(Context context, Bitmap bitmap) {
    ByteArrayOutputStream bytes = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 100, bytes);
    String path = MediaStore.Images.Media.insertImage(context.getContentResolver(), bitmap,  title: "Title",  description: null);
    return Uri.parse(path);
}
```

Retrieve updated information from firebase

Browse Course




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

    editCourse = findViewById(R.id.btnEditCourse);
    btnPreviewChapters = findViewById(R.id.btnPreviewChapters);
    btnEditChapters = findViewById(R.id.btnEditChapters);
    btnEditFlashcard = findViewById(R.id.btnEditFlashcard);
    btnAddDiscussion = findViewById(R.id.btnAddDiscussion);
    btnEditDiscussion = findViewById(R.id.btnEditDiscussion);
    btnEnrol = findViewById(R.id.btnEnrol);

    setupButton();

    yangding14
    editCourse.setOnClickListener(new View.OnClickListener() {
        yangding14
        @Override
        public void onClick(View view) { openEditCourse(); }
    });


    courseTitleTextView = findViewById(R.id.TVCourseTitle);
    courseDescTextView = findViewById(R.id.TVDesc);

    // Reference to the Firebase "Edit Course" section where instructor data is stored
    DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference( path: "Course");

// Attach a ValueEventListener to retrieve data from Firebase
yangding14
databaseRef.addValueEventListener(new ValueEventListener() {
    yangding14
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        // Check if data exists
        if (dataSnapshot.exists()) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                // Retrieve data for each course
                String courseTitle = snapshot.child( path: "courseTitle").getValue(String.class);
                String courseDesc = snapshot.child( path: "courseDesc").getValue(String.class);

                // Update the TextViews in the student view with the retrieved data
                courseTitleTextView.setText(courseTitle);
                courseDescTextView.setText(courseDesc);

            }
        }
    }
    yangding14
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});

courseImageView = findViewById(R.id.ImageCourse);

// Reference to the Firebase "Image" section where instructor data is stored
DatabaseReference dbRef = FirebaseDatabase.getInstance().getReference( path: "Course");

// Attach a ValueEventListener to retrieve data from Firebase
yangding14 +1
dbRef.addValueEventListener(new ValueEventListener() {
    yangding14 +1
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        // Check if data exists
        if (dataSnapshot.exists()) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                // Retrieve image
                String imageUrl = snapshot.child( path: "uri").getValue(String.class); // Retrieve image URL

                // Load the image using Glide or Picasso (you may need to add the corresponding libraries)
                if (imageUrl != null && !imageUrl.isEmpty()) {
                    Glide.with( activity: InCourseViewActivity.this) RequestManager
                            .load(imageUrl) RequestBuilder<Drawable>
                            .into(courseImageView);
                }
            }
        }
    }
```

Student's homepage

Browse the discover course and ongoing course

```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    // Recycle View for Discover Courses
    RecyclerView recyclerView = view.findViewById(R.id.discoverCourseRecycleView);
    setUpDiscoverCourseData();
    //DiscoverCourseAdapter discoverCourseAdapter = new DiscoverCourseAdapter(this.getContext(), discoverCoursesArray);
    recyclerView.setAdapter(discoverCourseAdapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(this.getContext(), LinearLayoutManager.HORIZONTAL, reverseLayout: false));

    // Recycle View for Ongoing Courses
    RecyclerView recyclerView1 = view.findViewById(R.id.ongoingCourseRecycleView);
    setUpOngoingCourseData();
    // OngoingCourseAdapter ongoingCourseAdapter = new OngoingCourseAdapter(this.getContext(), ongoingCourseArrayList);
    recyclerView1.setAdapter(ongoingCourseAdapter);
    recyclerView1.setLayoutManager(new LinearLayoutManager(this.getContext(), LinearLayoutManager.HORIZONTAL, reverseLayout: false));
```

```
public void setUpDiscoverCourseData(){
    String[] courseName = getResources().getStringArray(R.array.courses);
    String[] courseDesc = getResources().getStringArray(R.array.course_desc);

    for (int i=0; i<courseDesc.length; i++){
        discoverCoursesArray.add(new DiscoverCourse(courseName[i], courseDesc[i], courseImages[i]));
    }
}
```

```
1 usage   yangding14
public void setUpOngoingCourseData(){
    String[] oc_courseNames = getResources().getStringArray(R.array.oc_courseName);
    String[] oc_courseQtys = getResources().getStringArray(R.array.oc_courseQty);
    for (int i=0; i<oc_courseNames.length; i++){
        ongoingCourseArrayList.add(new OngoingCourse(oc_courseNames[i], oc_courseQtys[i], oc_courseImages[i]));
    }
}
```

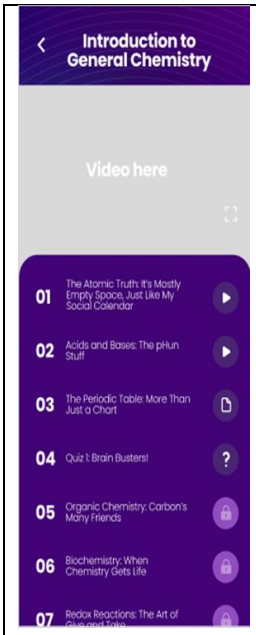
Revise the content of completed course

```
9 usages   yangding14 +1
public class StCourseViewActivity extends AppCompatActivity {
    2 usages
    private TextView courseTitleTextView, courseDescTextView;
    2 usages
    private Button btnStartStudy, btnRevision, btnForum;

    yangding14
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.course_st_course_view);

        courseTitleTextView = findViewById(R.id.TVCourseTitle);
        courseDescTextView = findViewById(R.id.TVDesc);

        btnStartStudy = findViewById(R.id.btnStartStudy);
        btnRevision = findViewById(R.id.btnRevision);
        btnForum = findViewById(R.id.btnForum);
        setupButton();

    private void setupButton() {
        yangding14
        btnStartStudy.setOnClickListener(new View.OnClickListener() {
            yangding14
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: StCourseViewActivity.this, com.example.eduempoweryd.videoview.MainActivity.class));
            }
        });

        yangding14
        btnRevision.setOnClickListener(new View.OnClickListener() {
            yangding14
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: StCourseViewActivity.this, com.example.eduempoweryd.flashcard.StudentActivity.class));
            }
        });

        yangding14 +1
        btnForum.setOnClickListener(new View.OnClickListener() {
            yangding14 +1
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: StCourseViewActivity.this, com.example.eduempoweryd.forum.AddComment.class));
            }
        });
    }
```

	To show learning progress `public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {` `super.onViewCreated(view, savedInstanceState);` `pieChart = view.findViewById(R.id.pieChart);` `DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference( path: "Chapters");` `coolkheng` `databaseReference.addValueEventListener(new ValueEventListener() {` `coolkheng` `@Override` `public void onDataChange(@NonNull DataSnapshot dataSnapshot) {` `progressChapterlists.clear(); // Clear existing data before adding new data` `int i = 0;` `for (DataSnapshot snapshot : dataSnapshot.getChildren()) {` `String name = snapshot.child( path: "name").getValue(String.class);` `String position = snapshot.child( path: "position").getValue(String.class);` `String key = snapshot.getKey();` `String filetype = snapshot.child( path: "file").getValue(String.class);` `if(images[i]!=0){` `progressChapterlists.add(new progress_chapterlist(position, name, image[0] , key));` `x++;` `}else {progressChapterlists.add(new progress_chapterlist(position, name, image[1] , key));` `y++;` `}` `i++;` `}` `showStatistics(x,y);` `}` `@Override` `public void onCancelled(@NonNull DatabaseError databaseError) {` `// Handle the error` `}` `});` **To show statistic of last accessed activity using pie chart** `private void showStatistics(int done , int not_done) {` `pieChart.setVisibility(View.VISIBLE);` `pieChart.getDescription().setEnabled(false);` `pieChart.setDrawHoleEnabled(true);` `List<PieEntry> entries = new ArrayList<>();` `entries.add(new PieEntry(done, label: ""));` `entries.add(new PieEntry(not_done, label: ""));` `PieDataSet set = new PieDataSet(entries, label: "");` `set.setColors(new int[] {` `ContextCompat.getColor(requireContext(), R.color.done),` `ContextCompat.getColor(requireContext(), R.color.not_done)` `});` `int percentage = done * 100 /(done+not_done);` `pieChart.setCenterText(String.valueOf(percentage)+"%");` `PieData data = new PieData(set);` `pieChart.setData(data);` `pieChart.invalidate(); // Refresh the pie chart` `// Optional: Add animation` `pieChart.animateXY( durationMillisX: 1400, durationMillisY: 1400);` `}` `}`

4.3.3 Class Material Module

Functional Requirements	**Source Code**
Study and manage class material	To retrieve the video or pdf file from the database `DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference("Chapters");`




```
databaseReference.addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
        chapterlists.clear(); // Clear existing data before
adding new data

        for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
            String name =
snapshot.child("name").getValue(String.class);
            String position =
snapshot.child("position").getValue(String.class);
            String key = snapshot.getKey();
            String filetype =
snapshot.child("file").getValue(String.class);




            if (!filetype.equals(null) &&
!filetype.trim().isEmpty()) {
                if
(getfiletype(Uri.parse(filetype)).equals("mp4")) {
                    chapterlists.add(new chapterlist(position,
name, images[0], key));
                } else if
(getfiletype(Uri.parse(filetype)).equals("pdf")) {
                    chapterlists.add(new chapterlist(position,
name, images[1], key));
                }
            } else {
                chapterlists.add(new chapterlist(position,
name, images[2], key));
            }


        }

        Collections.sort(chapterlists, new
Comparator<chapterlist>() {
            @Override
            public int compare(chapterlist chapter1,
chapterlist chapter2) {
                // Parse "position" as integers and compare
                int position1 =
```

```
Integer.parseInt(chapter1.getPosition());
                int position2 =
Integer.parseInt(chapter2.getPosition());

                // Compare the parsed integers
                return Integer.compare(position1, position2);
            }
        });
        chapteradapter.notifyDataSetChanged();


    }

    @Override
    public void onCancelled(@NonNull DatabaseError
databaseError) {
        // Handle the error
    }
```

To categorize the file to video file or pdf file

```
private String getfiletype(Uri fileuri) {

    List<String> pathSegments = fileuri.getPathSegments();
    // Get the last segment which contains the file name
    String fileName = pathSegments.get(pathSegments.size() - 1);

    // Find the last occurrence of '.' to get the file extension
    int dotIndex = fileName.lastIndexOf('.');

    // Get the substring after the last dot to get the file
extension
    String fileExtension = fileName.substring(dotIndex + 1);

    if(fileExtension.equals("mp4")) {
        return "mp4";
    }
    else{return "pdf";}
}
```

To display the video or the pdf file

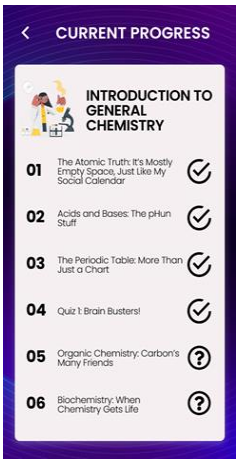
```
holder.imageButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
        String keyToPlay =
ChapterarrayList.get(adapterPosition).getKey();
        mediaController = new MediaController(v.getContext());
        DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("Chapters").child(keyTo
Play);
        databaseReference.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {

                if
(ChapterarrayList.get(adapterPosition).getImage() ==
R.drawable.play) {
                    String video =
snapshot.child("file").getValue(String.class);
                    videoView.setMediaController(mediaController);
                    mediaController.setAnchorView(videoView);
                    Uri videouri = Uri.parse(video);
                    videoView.setVideoURI(videouri);
                    videoView.start();

                } else if
(ChapterarrayList.get(adapterPosition).getImage() ==
R.drawable.file) {

                    String video =
snapshot.child("file").getValue(String.class);
                    Uri pdf = Uri.parse(video);

                    Intent intent = new Intent(Intent.ACTION_VIEW);
                    intent.setDataAndType(pdf, "application/pdf");

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    try{
                        v.getContext().startActivity(intent);
                    }catch(ActivityNotFoundException e){}
                }

            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
```

```
    }
});
```

Progress tracker

RecyclerView showing the progress



Pie-chart shows the progress of the student

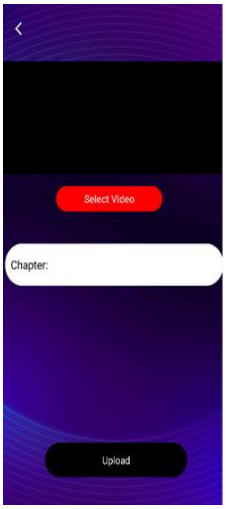
```
private void showStatistics(int done , int not_done) {
        pieChart.setVisibility(View.VISIBLE);

        pieChart.getDescription().setEnabled(false);
        pieChart.setDrawHoleEnabled(true);

        List<PieEntry> entries = new ArrayList<>();

        entries.add(new PieEntry(done, ""));
        entries.add(new PieEntry(not_done, ""));

        PieDataSet set = new PieDataSet(entries, "");
        set.setColors(new int[] {
                ContextCompat.getColor(requireContext(),
R.color.done),
                ContextCompat.getColor(requireContext(),
R.color.not_done)
        });

        int percentage = done * 100 /(done+not_done);
        pieChart.setCenterText(String.valueOf(percentage)+"%");




        PieData data = new PieData(set);
        pieChart.setData(data);
        pieChart.invalidate(); // Refresh the pie chart

        // Optional: Add animation
        pieChart.animateXY(1400, 1400);
    }
}
```

Instructors manage class material

Browse user's device storage

```
selectVideo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setType("video/*");
```

Add new chapter (video file)




```
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent, 101);
    }
});
```

Add new chapter to firebase storage (video file)

```
Button Upload_button = view.findViewById(R.id.upload_button);
Upload_button.setOnClickListener(new View.OnClickListener() {


    @Override
    public void onClick(View v) {
        Chapter=view.findViewById(R.id.ChaptereditText);
        String chapterText = Chapter.getText().toString();

        progressDialog = new ProgressDialog(requireContext());

        if (video != null) {
                // save the selected video in Firebase storage
                final StorageReference reference =
FirebaseStorage.getInstance().getReference("Files/" +
System.currentTimeMillis() + "." + getfiletype(video));
                reference.putFile(video).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
                        Task<Uri> uriTask =
taskSnapshot.getStorage().getDownloadUrl();
                        while (!uriTask.isSuccessful()) ;
                        // get the link of video
                        String downloadUri =
uriTask.getResult().toString();
                        DatabaseReference reference1 =
FirebaseDatabase.getInstance().getReference("Chapters");
                        UploadData("Chapters" , chapterText ,
downloadUri);
                        // Video uploaded successfully
                        // Dismiss dialog
                        progressDialog.dismiss();
                        Toast.makeText(requireContext(), "Video
Uploaded!!", Toast.LENGTH_SHORT).show();
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        // Error, Image not uploaded
                        progressDialog.dismiss();
```

```
                        Toast.makeText(requireContext(), "Failed "
+ e.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                }).addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
                    // Progress Listener for loading
                    // percentage on the dialog box
                    @Override
                    public void onProgress(UploadTask.TaskSnapshot
taskSnapshot) {
                        // show the progress bar
                        double progress = (100.0 *
taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                        progressDialog.setMessage("Uploaded " +
(int) progress + "%");
                    }
                });
            }else if (video==null){
            Toast.makeText(requireContext(), "Please upload video!
", Toast.LENGTH_SHORT).show();
        }



        }

    });
};
```

Upload the file link into the database

```
public void UploadData(String parentnode , String chapter_data ,
String Videolink ){
        DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference(parentnode);
        int passout ;
        databaseReference.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
                long numberOfKeys =
dataSnapshot.getChildrenCount();
                String string_position =
String.valueOf(numberOfKeys+1);


                if (string_position.length() == 1){
```

```
                    string_position = "0" + string_position;
                }


                // Now, numberOfKeys contains the count of child
nodes under "Chapters"
                // We get the String value of numberofKeys
                HashMap<String, String> Videomap = new
HashMap<>();
                Videomap.put("file", Videolink);
                Videomap.put("name" , chapter_data);
                Videomap.put("position",string_position);
                chapterlist data = new
chapterlist(string_position,chapter_data,Videolink);
//                HashMap<String, Object> dataMap = new
HashMap<>();
//                dataMap.put("", data);


                databaseReference.child("Chapter " +
String.valueOf(numberOfKeys+1) ).setValue(Videomap);




            }
```

Add new chapter (pdf file)

Add new chapter (pdf file)


```
Button Upload_button = view.findViewById(R.id.upload_button);
Upload_button.setOnClickListener(new View.OnClickListener() {


    @Override
    public void onClick(View v) {
        Chapter=view.findViewById(R.id.ChaptereditText);
        String chapterText = Chapter.getText().toString();


        progressDialog = new ProgressDialog(requireContext());


        if (pdf != null) {
            // save the selected video in Firebase storage
            final StorageReference reference =
FirebaseStorage.getInstance().getReference("Files/" +
System.currentTimeMillis() + "." + getfiletype(pdf));
            reference.putFile(pdf).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
```

```
                    Task<Uri> uriTask =
taskSnapshot.getStorage().getDownloadUrl();
                    while (!uriTask.isSuccessful()) ;
                    // get the link of video
                    String downloadUri =
uriTask.getResult().toString();
                    DatabaseReference reference1 =
FirebaseDatabase.getInstance().getReference("Chapters");
                    UploadData("Chapters" , chapterText ,
downloadUri);
                    // Video uploaded successfully
                    // Dismiss dialog
                    progressDialog.dismiss();
                    Toast.makeText(requireContext(), "Video
Uploaded!!", Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    // Error, Image not uploaded
                    progressDialog.dismiss();
                    Toast.makeText(requireContext(), "Failed " +
e.getMessage(), Toast.LENGTH_SHORT).show();
                }
            }).addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
                // Progress Listener for loading
                // percentage on the dialog box
                @Override
                public void onProgress(UploadTask.TaskSnapshot
taskSnapshot) {
                    // show the progress bar
                    double progress = (100.0 *
taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                    progressDialog.setMessage("Uploaded " + (int)
progress + "%");
                }
            });
        }else if (pdf==null){
            Toast.makeText(requireContext(), "Please upload video!
", Toast.LENGTH_SHORT).show();
        }



    }

});
```

```
        });
    }
```

Edit chapter (video file & pdf file)

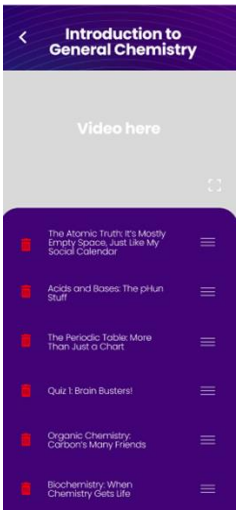
```
Button select_button = getView().findViewById(R.id.select_button);
select_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setType("video/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(intent, 101);

        progressDialog = new ProgressDialog(requireContext());

        if (video != null) {
            // save the selected video in Firebase storage
            final StorageReference reference =
FirebaseStorage.getInstance().getReference("Files/" +
System.currentTimeMillis() + "." + getfiletype(video));
            reference.putFile(video).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
                    Task<Uri> uriTask =
taskSnapshot.getStorage().getDownloadUrl();
                    while (!uriTask.isSuccessful()) ;
                    // get the link of video
                    String downloadUri =
uriTask.getResult().toString();
                    DatabaseReference reference1 =
FirebaseDatabase.getInstance().getReference("Chapters");

databaseReference.child(list.get(position).getKey()).child("file").
setValue(downloadUri);
                    // Video uploaded successfully
                    // Dismiss dialog
                    progressDialog.dismiss();
                    Toast.makeText(requireContext(), "Video
Uploaded!!", Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
```

```
                    // Error, Image not uploaded
                    progressDialog.dismiss();
                    Toast.makeText(requireContext(), "Failed " +
e.getMessage(), Toast.LENGTH_SHORT).show();
                }
            }).addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
                // Progress Listener for loading
                // percentage on the dialog box
                @Override
                public void onProgress(UploadTask.TaskSnapshot
taskSnapshot) {
                    // show the progress bar
                    double progress = (100.0 *
taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                    progressDialog.setMessage("Uploaded " + (int)
progress + "%");
                }
            });
        } else if (video == null) {
            Toast.makeText(requireContext(), "Please upload video!
", Toast.LENGTH_SHORT).show();
        }



    }
});
```

Check if the chapter name is changed

```
private boolean isChapterNameChanged() {
    String ChapterName = ChapterarrayList.get(position).getName();
    String File = ChapterarrayList.get(position).getFileType();

    if (!ChapterName.equals(ChapterText.getText().toString())){

databaseReference.child(ChapterarrayList.get(position).getKey()).c
hild("name").setValue(ChapterText.getText().toString());
        return true;
    } else {
        return false;
    }
}
```

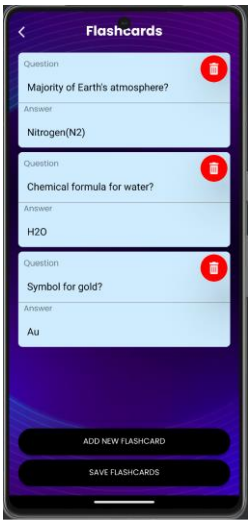
Delete chapter from database

Delete chapter 	holder.delete_button.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View v) { AlertDialog.Builder builder =new AlertDialog.Builder(holder.tvName.getContext()); builder.setTitle("Are you sure?"); builder.setMessage("Deleted data can't be undo"); builder.setPositiveButton("Delete", new DialogInterface.OnClickListener() { @Override public void onClick(DialogInterface dialog, int which) { String keyToDelete = editChapter2arrayList.get(adapterPosition).getKey(); FirebaseDatabase.*getInstance*().getReference().child("Chapters") .child(keyToDelete).removeValue(); } }); builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() { @Override public void onClick(DialogInterface dialog, int which) { Toast.*makeText*(holder.tvName.getContext() , "Cancelled" , Toast.*LENGTH_SHORT*).show(); } }); builder.show(); } });

4.3.4 Flashcard Module

Functional Requirements	Source Code

Manage flashcards



To retrieve flashcard data from the database

```
mDatabase.child("flashcards").addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        flashcardsList.clear(); // Clear the old list
        for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
            Flashcard flashcard =
snapshot.getValue(Flashcard.class);
            flashcardsList.add(flashcard);
        }
        adapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Toast.makeText(InstructorActivity.this, "Failed to
load flashcards.", Toast.LENGTH_SHORT).show();
    }
});
```

To display the flashcards

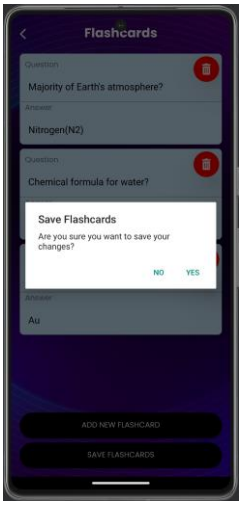
```
private class FlashcardAdapter extends BaseAdapter {

    @Override
    public int getCount() {
        return flashcardsList.size();
    }

    @Override
    public Flashcard getItem(int position) {
        return flashcardsList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView,
ViewGroup parent) {
        if (convertView == null) {
            convertView =
LayoutInflater.from(InstructorActivity.this).inflate(R.layo
ut.flashcard_list_item_flashcard, parent, false);
```

```
        }

        EditText etQuestion =
convertView.findViewById(R.id.et_question);
        EditText etAnswer =
convertView.findViewById(R.id.et_answer);
        ImageButton btnDelete =
convertView.findViewById(R.id.btn_delete);

        Flashcard flashcard = getItem(position);
        etQuestion.setText(flashcard.getQuestion());
        etAnswer.setText(flashcard.getAnswer());

        etQuestion.setOnFocusChangeListener(new
View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View view, boolean
hasFocus) {
                if (!hasFocus) {

flashcard.setQuestion(etQuestion.getText().toString());
                }
            }
        });

        etAnswer.setOnFocusChangeListener(new
View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View view, boolean
hasFocus) {
                if (!hasFocus) {

flashcard.setAnswer(etAnswer.getText().toString());
                }
            }
        });
```

To add new flashcard to the list or delete existing flashcard from the list

```
btnAddFlashcard.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        flashcardsList.add(new Flashcard("", ""));
        adapter.notifyDataSetChanged();
    }
});
btnDelete.setOnClickListener(new View.OnClickListener() {
    @Override
```

Save changes




```
    public void onClick(View view) {
        flashcardsList.remove(position);
        notifyDataSetChanged();
    }
});

return convertView;
```

To save the changes

```
btnSaveFlashcards.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        showSaveConfirmation();
    }
});

private void saveFlashcards() {
    mDatabase.child("flashcards").setValue(flashcardsList);
    Toast.makeText(this, "Flashcards saved successfully!",
Toast.LENGTH_SHORT).show();
}

private void showSaveConfirmation() {
    AlertDialog.Builder builder = new
AlertDialog.Builder(InstructorActivity.this);
    builder.setTitle("Save Flashcards");
    builder.setMessage("Are you sure you want to save your
changes?");

    builder.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id)
{
            saveFlashcards();
        }
    });

    builder.setNegativeButton("No", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id)
{
            dialog.dismiss();
        }
    });

    AlertDialog dialog = builder.create();
```

```
    dialog.show();
}
```

Browse flashcards

To retrieve flashcard data from database

```
// Firebase Realtime Database listener
        mDatabase.addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot
dataSnapshot) {
                flashcardsList.clear();
                for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                    Flashcard flashcard =
snapshot.getValue(Flashcard.class);
                    flashcardsList.add(flashcard);
                }
                if (!flashcardsList.isEmpty()) {
                    setupFlashcardView();
                }
            }

            @Override
            public void onCancelled(DatabaseError
databaseError) {
//                Log.e("StudentActivity", "Error fetching
data", databaseError.toException());
                System.out.println("The read failed: " +
databaseError.getCode());
            }
        });
```

To set up the flashcard view

```
private void setupFlashcardView() {
    if (flashcardsList != null
&& !flashcardsList.isEmpty()) {
        Flashcard currentFlashcard =
flashcardsList.get(currentCardIndex);

questionTextView.setText(currentFlashcard.getQuestion());

answerTextView.setText(currentFlashcard.getAnswer());

        // Initially hide the answer and buttons until the
user asks to show the answer
        answerTextView.setVisibility(View.GONE);
```

Rate flashcard	 ``` easyButton.setVisibility(View.GONE); goodButton.setVisibility(View.GONE); showAnswerButton.setVisibility(View.VISIBLE); } else { questionTextView.setText("No flashcards available."); showAnswerButton.setEnabled(false); } } private void showNextCard() { if (flashcardsList != null && !flashcardsList.isEmpty() && currentCardIndex < flashcardsList.size() - 1) { currentCardIndex++; setupFlashcardView(); } else { questionTextView.setVisibility(View.GONE); answerTextView.setVisibility(View.GONE); easyButton.setVisibility(View.GONE); goodButton.setVisibility(View.GONE); showAnswerButton.setVisibility(View.GONE); congratsAndChart.setVisibility(View.VISIBLE); congratsTextView.setVisibility(View.VISIBLE); againButton.setVisibility(View.VISIBLE); showStatistics(); } } ``` To check the answer ``` showAnswerButton.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View view) { answerTextView.setVisibility(View.VISIBLE); easyButton.setVisibility(View.VISIBLE); goodButton.setVisibility(View.VISIBLE); showAnswerButton.setVisibility(View.GONE); } }); ``` To rate their understanding ``` easyButton.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View view) { easyCount++; showNextCard(); ```


```
    }
});

goodButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        goodCount++;
        showNextCard();
    }
});
```

Flashcard recommendation

To reset the flashcard

```
againButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        resetFlashcards();
    }
});
private void resetFlashcards() {
    currentCardIndex = 0;
    easyCount = 0;
    goodCount = 0;

    setupFlashcardView();
    congratsTextView.setVisibility(View.GONE);
    questionTextView.setVisibility(View.VISIBLE);
    showAnswerButton.setVisibility(View.VISIBLE);
    againButton.setVisibility(View.GONE);
    pieChart.setVisibility(View.GONE);
    congratsAndChart.setVisibility(View.GONE);
}
```

View statistics

To show statistics of the performance

```
private void showStatistics() {
    pieChart.setVisibility(View.VISIBLE);

    pieChart.getDescription().setEnabled(false);
    pieChart.setDrawHoleEnabled(true);

    List<PieEntry> entries = new ArrayList<>();

    entries.add(new PieEntry(easyCount, "Easy"));
    entries.add(new PieEntry(goodCount, "Good"));
```

	`` PieDataSet set = new PieDataSet(entries, ""); set.setColors(new int[] { ContextCompat.getColor(this, R.color.pie_easy), ContextCompat.getColor(this, R.color.pie_good) }); PieData data = new PieData(set); pieChart.setData(data); pieChart.invalidate(); // Refresh the pie chart // Add animation pieChart.animateXY(1400, 1400); } ```

4.3.5 Quiz Module

Functional Requirements	Source Code
Attempt quiz and review quiz  Attempt quiz	**Attempt quiz** To get the question from database ``` // fetch data from database and populate questions along with all options into the questions list public void awaitPopulateQuestion(OnQuestionsPopulatedListener listener) { Log.d("awaitPopulateQuestion", "Populating questions"); // calling add value event listener method for getting the values from the database. db.child("questions").addListenerForSingleValueEvent(new ValueEventListener() { @Override public void onDataChange(@NonNull DataSnapshot snapshot) { questions.clear(); // Clear existing questions Log.d("awaitPopulateQuestion", "Snapshot: " + snapshot.toString()); for (DataSnapshot questionSnapshot : snapshot.getChildren()) { // Get the values from the snapshot ```

```
                String questionId =
questionSnapshot.getKey();
                String questionText =
questionSnapshot.child("question").getValue(String.class);
                Log.d("awaitPopulateQuestion",
"questionSnapshot: " + questionSnapshot.toString());

                // Get the options, which is a list of
strings
                List<String> options = new ArrayList<>();
                for (DataSnapshot optionSnapshot :
questionSnapshot.child("options").getChildren()) {
                    String option =
optionSnapshot.getValue(String.class);
                    options.add(option);
                }

                // Create a Question object and add it to
the questions list
                Question question = new Question();
                question.setQuestionId(questionId);
                question.setQuestion(questionText);
                question.setOptions(options);

                // Add the question to the questions list
                questions.add(question);
            }


            Log.d("awaitPopulateQuestion", "Questions: " +
questions.get(0).getQuestion() + " " +
questions.get(0).getOptions().toString());

            // Notify the listener that questions are
populated
            if (listener != null) {
                listener.onQuestionsPopulated();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError
error) {
            // calling on cancelled method when we receive
any error or we are not able to get the data.
            Log.e("StQuizQuestionsList", "Fail to get
data.");
        }
```

```
    });
}
```

After fetching data, display the questions and options

```
// fetch data from database and populate questions along
with all options into the questions list
public void
awaitPopulateQuestion(OnQuestionsPopulatedListener
listener) {
    Log.d("awaitPopulateQuestion", "Populating questions");
    // calling add value event listener method for getting
the values from the database.

db.child("questions").addListenerForSingleValueEvent(new
ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            questions.clear(); // Clear existing questions
            Log.d("awaitPopulateQuestion", "Snapshot: " +
snapshot.toString());

            for (DataSnapshot questionSnapshot :
snapshot.getChildren()) {
                // Get the values from the snapshot
                String questionId =
questionSnapshot.getKey();
                String questionText =
questionSnapshot.child("question").getValue(String.class);
                Log.d("awaitPopulateQuestion",
"questionSnapshot: " + questionSnapshot.toString());

                // Get the options, which is a list of
strings
                List<String> options = new ArrayList<>();
                for (DataSnapshot optionSnapshot :
questionSnapshot.child("options").getChildren()) {
                    String option =
optionSnapshot.getValue(String.class);
                    options.add(option);
                }

                // Create a Question object and add it to
the questions list
                Question question = new Question();
                question.setQuestionId(questionId);
                question.setQuestion(questionText);
                question.setOptions(options);
```

```
                // Add the question to the questions list
                questions.add(question);
            }


            Log.d("awaitPopulateQuestion", "Questions: " +
questions.get(0).getQuestion() + " " +
questions.get(0).getOptions().toString());

            // Notify the listener that questions are
populated
            if (listener != null) {
                listener.onQuestionsPopulated();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError
error) {
            // calling on cancelled method when we receive
any error or we are not able to get the data.
            Log.e("StQuizQuestionsList", "Fail to get
data.");
        }
    });
}
```

When the user clicks on any answer

```
public void clickOnAnswer(View v){
    TextView clickedTextView = (TextView) v;
    // reset timer everytime enter a new question
    countdown.cancel();

    // Get the question
    Question question = questions.get(currentQuestion);

    // Get the answer
    String answer = clickedTextView.getText().toString();

    // Get the correct answer
    String correctAnswer = question.getOptions().get(0);

    // Check if the answer is correct
    boolean result = answer.equals(correctAnswer);
    Log.d("InQuizAddQuestionFrag", "question: " +
question.getQuestion());
```

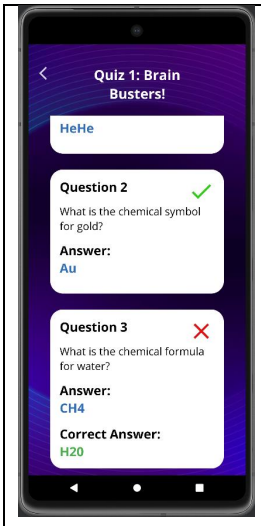
```
    // Add the question, answer and result to the attempt
    attempt.setQuestion(question.getQuestion());
    attempt.setChosenAnswer(answer);
    attempt.setCorrectAnswer(correctAnswer);
    attempt.setResult(result);

    // Store the attempt in the database
    storeAttemptToFirebase(attempt);

    // Check if the current question is the last question
    if (currentQuestion == questions.size() - 1) {
        // End the quiz
        endQuiz();
    } else {
        // Increment the current question
        currentQuestion++;

        // Display the next question
        displayQuestion();
        displayOptions();
    }
}

// Store the attempt in the database under the attempt key
private void storeAttemptToFirebase(QuestionAttempt
attempt){
    Bundle bundle = this.getArguments();
    String attemptKey = bundle.getString("attemptKey");

    // Use setValue on the specific reference

db.child("attempts").child(attemptKey).child("user_attempts
").push().setValue(attempt)
            .addOnSuccessListener(aVoid -> {
                Log.d("StQuizInsideQuizFrag", "Attempt
stored successfully");
            })
            .addOnFailureListener(e -> {
                // Show error message
                Toast.makeText(getActivity(), "Error: " +
e.getMessage(), Toast.LENGTH_LONG).show();
            });

}
```

View quiz result

View quiz result

```
private void displayResult(){
    Bundle bundle = this.getArguments();
```

```
    String attemptKey = bundle.getString("attemptKey");


db.child("quizzes").child("quiz1").child("attempts").child(
attemptKey)
            .addListenerForSingleValueEvent(new
ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            int questionSize = (int)
snapshot.child("questionSize").getValue(Integer.class);
            int attemptSize = (int)
snapshot.child("user_attempts").getChildrenCount();
            int correctSize = 0;

            for (DataSnapshot attemptSnapshot :
snapshot.child("user_attempts").getChildren()) {
                // Get the values from the snapshot
                Boolean result =
attemptSnapshot.child("result").getValue(Boolean.class);
                if (result) {
                    correctSize++;
                }
            }

            bigResult.setText(correctSize + "/" +
questionSize);

completionsValue.setText((int)((double)attemptSize/(double)
questionSize*100) + "%");
            questionsValue.setText(questionSize + "");
            correctValue.setText(correctSize + "");
            wrongValue.setText((questionSize - correctSize)
+ "");
        }

        @Override
        public void onCancelled(@NonNull DatabaseError
error) {
            // calling on cancelled method when we receive
any error or we are not able to get the data.
            Log.e("StQuizResultFragment", "Fail to get
data.");
        }
    });
}
```



Review answers

Review answers

```
public void populateAttempts(){
    Bundle bundle = this.getArguments();
    String attemptKey = bundle.getString("attemptKey");


db.child("quiz1").child("attempts").child(attemptKey).child
("user_attempts").addValueEventListener(new
ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            attempts.clear(); // Clear existing questions

            for (DataSnapshot questionSnapshot :
snapshot.getChildren()) {
                // Get the values from the snapshot
                String question =
questionSnapshot.child("question").getValue(String.class);
                String chosenAnswer =
questionSnapshot.child("chosenAnswer").getValue(String.clas
s);
                String correctAnswer =
questionSnapshot.child("correctAnswer").getValue(String.cla
ss);
                Boolean result =
questionSnapshot.child("result").getValue(Boolean.class);

                // Create a QuestionAttempt object and add
it to the attempts list
                QuestionAttempt questionAttempt = new
QuestionAttempt();
                questionAttempt.setQuestion(question);

questionAttempt.setChosenAnswer(chosenAnswer);

questionAttempt.setCorrectAnswer(correctAnswer);
                questionAttempt.setResult(result);

                attempts.add(questionAttempt);
            }

            questionReviewAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError
```

```
error) {
            // calling on cancelled method when we receive
            // any error or we are not able to get the
data.
            Toast.makeText(getActivity(), "Fail to get
data.", Toast.LENGTH_SHORT).show();
        }
    });

}
```

Manage quiz

To add more selection

```
private void addSelection(){
    float scale =
getResources().getDisplayMetrics().density;
    EditText answerEditText = new
EditText(requireContext());

    LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT
    );

    // Set margins in pixels (convert dp to pixels)
    layoutParams.setMargins(0, 0, 0, dp(10));
    answerEditText.setLayoutParams(layoutParams);

    // Set styling for EditText
    answerEditText.setHint("Answer " +
(answerEditTextList.size() + 1));

answerEditText.setBackground(getResources().getDrawable(R.d
rawable.rounded_rectangle_white));
    answerEditText.setPadding(dp(30), dp(18), dp(30),
dp(18));
    linearAnswersContainer.addView(answerEditText);
    answerEditTextList.add(answerEditText);
}
```

To store the changes to database

```
private void storeDataToFirebase_updateData(){
    Log.d("InQuizAddQuestionFrag", "Save Changes button
pressed - update data");
    Bundle bundle = this.getArguments();
```

```
    // Code here executes on main thread after user presses
button
    String questionText = newQuestion.getText().toString();
    String correctAnswerText =
correctAnswer.getText().toString();

    // Get answers from the EditText views
    List<String> answers = new ArrayList<>();
    for (EditText answerEditText : answerEditTextList) {
        answers.add(answerEditText.getText().toString());
    }

    //check whether the edittext fields are empty
    if (TextUtils.isEmpty(questionText) ||
TextUtils.isEmpty(correctAnswerText)){
        Toast.makeText(getActivity(), "Please fill in all
fields", Toast.LENGTH_LONG).show();
    }else if(answers.size() == 0 ||
TextUtils.isEmpty(answers.get(0))){
        Toast.makeText(getActivity(), "Please add at least
one answer", Toast.LENGTH_LONG).show();
    }else{
        //add the question to the database
        Log.d("InQuizAddQuestionFrag", "Storing data to
firebase");

        List<String> options = new ArrayList<>();
        options.add(correctAnswerText);
        options.addAll(answers);

        //set data in our object class
        question.setQuestion(questionText);
        question.setOptions(options);

        // Use setValue on the specific reference

dbRef.child(bundle.getString("questionId")).setValue(questi
on)
                .addOnSuccessListener(aVoid -> {
                    Toast.makeText(getActivity(), "Question
updated successfully", Toast.LENGTH_LONG).show();
                    Log.d("InQuizAddQuestionFrag",
"Question updated successfully");
                    switchFragment();
                })
                .addOnFailureListener(e -> {
                    // Show error message
                    Toast.makeText(getActivity(), "Error: "
```

```
+ e.getMessage(), Toast.LENGTH_LONG).show();
                });
    }
}
```

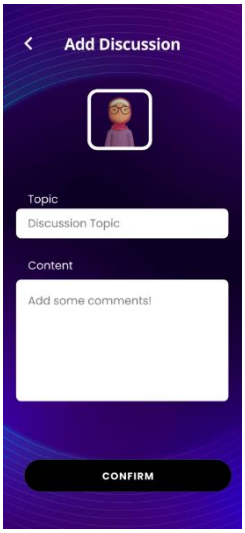
Analyze quiz

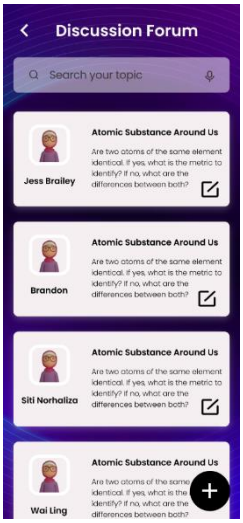
To setup the view

```
private void setupView(){
    // Assuming you have a LinearLayout in your XML layout
file with the id llQuizHistoryContainer
    LinearLayout quizHistoryContainer =
getView().findViewById(R.id.questionList);

    // Loop through the data lists and create views for
each quiz history entry
    for (int i = 0; i < date.size(); i++) {
        // Inflate the layout for a single quiz history
entry
        View historyEntryView =
getLayoutInflater().inflate(R.layout.quiz_history_entry,
quizHistoryContainer, false);

        // Find the TextViews in the inflated layout
        TextView attemptDate =
historyEntryView.findViewById(R.id.attemptDate);
        TextView attemptScore =
historyEntryView.findViewById(R.id.attemptScore);
        TextView attemptNumber =
historyEntryView.findViewById(R.id.attemptNumber);

        // Set the data to the TextViews
        attemptDate.setText(date.get(i));
        attemptScore.setText(score.get(i));
        attemptNumber.setText("Attempt " + String.valueOf(i
+ 1));

        // Set an OnClickListener for the ConstraintLayout
        final int position = i;  // final variable to be
used inside onClick
        historyEntryView.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Handle the click event for the clicked
ConstraintLayout
                onQuizHistoryEntryClick(position);
            }
        });
```

```
        // Add the history entry view to the container
        quizHistoryContainer.addView(historyEntryView);
    }
}
```

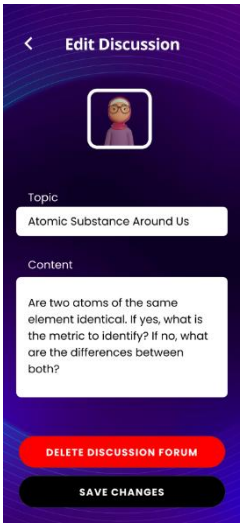
4.3.6 Discussion Module

Functional Requirements	Source Code
Create discussion forum 	To add the topic and content of a new discussion forum `// Assuming you have a DatabaseReference reference` `DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference("Discussion");` `// Initialize EditText fields` `txtTopic = findViewById(R.id.txtTopicIn);` `txtContent = findViewById(R.id.txtContentIn);` `findViewById(R.id.btnConfirm).setOnClickListener(new View.OnClickListener() {` `@Override` `public void onClick(View view) {` `// Extract values from EditText fields` `String topic = txtTopic.getText().toString();` `String content = txtContent.getText().toString();` `// Check if the values are not empty before saving` `if (!topic.isEmpty() && !content.isEmpty()) {` `// Create a new DiscussionItem object with extracted data` `// DiscussionItem discussionItem = new DiscussionItem(topic, comment);` `String key = databaseReference.push().getKey();` `assert key != null;` `DiscussionItem discussionItem = new DiscussionItem(key, topic, content, null);` `// // Push the data to Firebase` `// String key = databaseReference.push().getKey();` `// assert key != null;` `databaseReference.child(key).setValue(discussionItem);` `CharSequence text = "Saved successfully!";` `int duration = Toast.LENGTH_SHORT;` `Toast toast = Toast.makeText(InAddForum.this, text, duration);` `toast.show();`

	`` // Optionally, clear the EditText fields after saving txtTopic.setText(""); txtContent.setText(""); // Navigate back to the InViewForum activity Intent intent = new Intent(InAddForum.this, InViewForum.class); startActivity(intent); } } }); ```
View Forum (Instructor's side)  	To view all the discussion forum ``` // Assuming you have a DatabaseReference reference DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference("Discussion"); // Assuming myRef points to the "discussionItems" node in your Firebase Database databaseReference.addValueEventListener(new ValueEventListener() { final RecyclerView recyclerView = findViewById(R.id.recycler); @Override public void onDataChange(@NonNull DataSnapshot dataSnapshot) { discussionItemList.clear(); for (DataSnapshot snapshot : dataSnapshot.getChildren()) { DiscussionItem discussionItem = snapshot.getValue(DiscussionItem.class); discussionItemList.add(discussionItem); } // Now you have the list of DiscussionItem objects, you can use it as needed. // For example, you can pass it to your RecyclerView adapter. DiscussionItemAdapter adapter = new DiscussionItemAdapter(InViewForum.this, discussionItemList, InViewForum.this); adapter.setStudentView(false); recyclerView.setAdapter(adapter); recyclerView.addOnItemTouchListener(new RecyclerViewClickListener(InViewForum.this, recyclerView, new RecyclerViewClickListener.OnItemClickListener() { @Override public void onItemClick(View view, int position) { // Handle item click, for example, launch a new ```

```
activity
                Intent intent = new Intent(InViewForum.this,
InEditForum.class);
                startActivity(intent);
            }
        }));



        // Set the itemCount in your adapter
        adapter.setItemCount(discussionItemList.size());

        searchView = findViewById(R.id.searchViewSearch);
        searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                List<DiscussionItem> filteredList = new
ArrayList<>();

                for (DiscussionItem item : discussionItemList) {
                    if
(item.getTopic().toLowerCase().contains(newText.toLowerCase())) {
                        filteredList.add(item);
                    }
                }

                adapter.updateList(filteredList);
                return true;
            }


        });
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
        CharSequence text = "Fail to get data!";
        int duration = Toast.LENGTH_SHORT;
        Toast toast = Toast.makeText(InViewForum.this, text,
duration);
        toast.show();
    }

});
```

Edit discussion



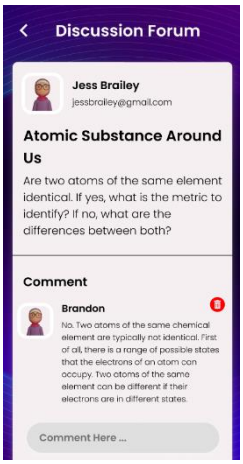

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.in_edit_forum);

    // Assuming you have a DatabaseReference reference
    DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("Discussion");

    // Initialize EditText fields
    txtTopic = findViewById(R.id.txtTopic);
    txtContent = findViewById(R.id.txtContent);

    // Retrieve the item key passed from the previous activity
    Bundle extras = getIntent().getExtras();
    if (extras != null) {
        itemKey = extras.getString("itemKey");
    }

    // Log the itemKey to verify its value
    Log.d("ItemKey", "Item Key: " + itemKey);

    // Assuming the itemKey is not null, you can use it to fetch the
specific DiscussionItem
    if (itemKey != null) {
        DatabaseReference specificItemRef =
databaseReference.child(itemKey);
        specificItemRef.get().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // Fetch the DiscussionItem corresponding to the
itemKey
                DiscussionItem discussionItem =
task.getResult().getValue(DiscussionItem.class);

                // Populate the EditText fields with the fetched
data
                if (discussionItem != null) {
                    txtTopic.setText(discussionItem.getTopic());
                    txtContent.setText(discussionItem.getContent());
                }
            }
        });
    }

    findViewById(R.id.btnSaveChanges).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
```

```
            // Extract values from EditText fields
            String topic = txtTopic.getText().toString();
            String content = txtContent.getText().toString();

            // Check if the values are not empty before saving
            if (!topic.isEmpty() && !content.isEmpty()) {
                DiscussionItem updatedDiscussionItem = new
DiscussionItem(itemKey, topic, content, null);
                // Create a new HashMap to update specific fields
                HashMap<String, Object> updateData = new
HashMap<>();
                updateData.put("topic", topic);
                updateData.put("content", content);

                // Update the data in Firebase using the itemKey

databaseReference.child(itemKey).setValue(updatedDiscussionItem);

                CharSequence text = "Changes saved successfully!";
                int duration = Toast.LENGTH_SHORT;
                Toast toast = Toast.makeText(InEditForum.this, text,
duration);
                toast.show();

                // Optionally, clear the EditText fields after
saving
                txtTopic.setText("");
                txtContent.setText("");

                Intent intent = new Intent(InEditForum.this,
InViewForum.class);
                startActivity(intent);
            }
        }
    });

    findViewById(R.id.btnDelete).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            LayoutInflater inflater = (LayoutInflater)
getSystemService(LAYOUT_INFLATER_SERVICE);
            View popupView =
inflater.inflate(R.layout.in_delete_forum, null);

            // create the popup window
            int width = LinearLayout.LayoutParams.WRAP_CONTENT;
            int height = LinearLayout.LayoutParams.WRAP_CONTENT;
```

```
            boolean focusable = true; // lets taps outside the popup
also dismiss it
            final PopupWindow popupWindow = new
PopupWindow(popupView, width, height, focusable);


            // show the popup window
            // which view you pass in doesn't matter, it is only
used for the window tolken

popupWindow.setAnimationStyle(R.style.popup_window_animation);
            popupWindow.showAtLocation(view, Gravity.CENTER, 0, 0);


            TextView txtDelete =
popupView.findViewById(R.id.txtDelete);
            txtDelete.setOnClickListener(new View.OnClickListener()
{
                @Override
                public void onClick(View v) {


                    // Check if the itemKey is not null before
attempting to delete
                    if (itemKey != null) {
                        // Assuming you have a DatabaseReference
reference
                        DatabaseReference specificItemRef =
databaseReference.child(itemKey);


                        // Remove the specific item from Firebase

specificItemRef.removeValue().addOnCompleteListener(task -> {
                                    if (task.isSuccessful()) {
                                        // Optionally, clear the
EditText fields after deleting
                                        txtTopic.setText("");
                                        txtContent.setText("");




                                    }
                                }
                        );



                        // Navigate back to the InViewForum activity
                        Intent intent = new Intent(InEditForum.this,
InViewForum.class);
                        startActivity(intent);
                        CharSequence text = "Discussion deleted
successfully!";
```

```
                        int duration = Toast.LENGTH_SHORT;
                        Toast toast =
Toast.makeText(InEditForum.this, text, duration);
                        toast.show();
                    }
                    popupWindow.dismiss();


                }
            });

            TextView txtSend =
popupView.findViewById(R.id.txtCancel);
            txtSend.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    popupWindow.dismiss();
                }

            });
        }
    });
}
```

Add Comment

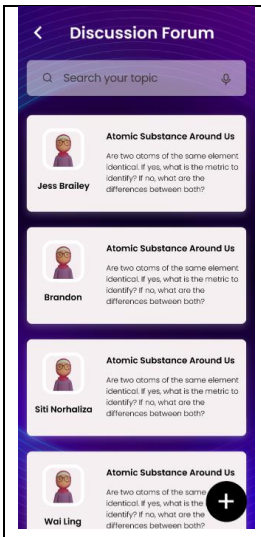


To add comment in particular discussion forum

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.add_comment);

    DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("Discussion");

    txtTopic = findViewById(R.id.txtTOPIC);
    txtContent = findViewById(R.id.txtCONTENT);
    txtComment = findViewById(R.id.comment);

    recyclerViewComments = findViewById(R.id.recyclerComments);
    LinearLayoutManager layoutManager = new
LinearLayoutManager(this);
    recyclerViewComments.setLayoutManager(layoutManager);

    Bundle extras = getIntent().getExtras();
    if (extras != null) {
        itemKey = extras.getString("itemKey");
    }

    Log.d("ItemKey", "Item Key: " + itemKey);
```

```
    if (itemKey != null) {
        DatabaseReference specificItemRef =
databaseReference.child(itemKey);
        specificItemRef.get().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                DiscussionItem discussionItem =
task.getResult().getValue(DiscussionItem.class);

                if (discussionItem != null) {
                    txtTopic.setText(discussionItem.getTopic());
                    txtContent.setText(discussionItem.getContent()
!= null ? discussionItem.getContent() : "No content available");
                    retrieveAllComments(specificItemRef);  //
Corrected line
                }
            }
        });
    }

    findViewById(R.id.addbutton).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String newComment =
txtComment.getText().toString().trim();

            if (!newComment.isEmpty()) {
                DatabaseReference specificItemRef =
databaseReference.child(itemKey).child("comments");
                String commentKey = specificItemRef.push().getKey();

                CommentItem comment = new CommentItem("name",
newComment);
                assert commentKey != null;
                specificItemRef.child(commentKey).setValue(comment);

                txtComment.setText("");  // Corrected line

                Toast.makeText(AddComment.this, "Comment added
successfully!", Toast.LENGTH_SHORT).show();
                // Now, fetch all comments for the specific item

retrieveAllComments(databaseReference.child(itemKey));
            } else {
                Toast.makeText(AddComment.this, "Please enter a
comment", Toast.LENGTH_SHORT).show();
            }
```

	} }); } private void retrieveAllComments(DatabaseReference specificItemRef) { specificItemRef.child("comments").get().addOnCompleteListener(task - > { if (task.isSuccessful()) { DataSnapshot dataSnapshot = task.getResult(); if (dataSnapshot.exists()) { List<CommentItem> commentItemList = new ArrayList<>(); for (DataSnapshot commentSnapshot : dataSnapshot.getChildren()) { CommentItem commentItem = commentSnapshot.getValue(CommentItem.class); if (commentItem != null) { commentItemList.add(0, commentItem); // Add new comments to the beginning of the list Log.d("Comment", "Author: " + commentItem.getAuthor() + ", Content: " + commentItem.getComment()); } } commentAdapter = new CommentAdapter(commentItemList); recyclerViewComments.setAdapter(commentAdapter); commentAdapter.notifyDataSetChanged(); // Notify adapter about the new data } } }); }
View Forum (Student's side)	protected void onCreate(@Nullable Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.*stud_view_forum*); // Assuming you have a DatabaseReference reference DatabaseReference databaseReference = FirebaseDatabase.*getInstance*().getReference("Discussion"); // Assuming myRef points to the "discussionItems" node in your Firebase Database databaseReference.addValueEventListener(new ValueEventListener() { final RecyclerView recyclerView =




```
findViewById(R.id.recycler);
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
            discussionItemList.clear();


            for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                DiscussionItem discussionItem =
snapshot.getValue(DiscussionItem.class);
                discussionItemList.add(discussionItem);
            }


            // Now you have the list of DiscussionItem objects, you
can use it as needed.
            // For example, you can pass it to your RecyclerView
adapter.
            DiscussionItemAdapter adapter = new
DiscussionItemAdapter(com.example.forum.students.StudViewForum.this,
discussionItemList, com.example.forum.students.StudViewForum.this);
            recyclerView.setAdapter(adapter);
            adapter.setStudentView(true);
            // Set click listener for the RecyclerView
            recyclerView.addOnItemTouchListener(new
RecyclerViewClickListener(StudViewForum.this, recyclerView, new
RecyclerViewClickListener.OnItemClickListener() {
                @Override
                public void onItemClick(View view, int position) {
                    // Handle item click, for example, launch a new
activity
                    DiscussionItem selectedDiscussionItem =
discussionItemList.get(position);


                    // Start the InEditForum activity and pass the
item key
                    Intent intent = new Intent(StudViewForum.this,
AddComment.class);
                    intent.putExtra("itemKey",
selectedDiscussionItem.getKey());
                    startActivity(intent);
                }
            }));



            // Set the itemCount in your adapter
            adapter.setItemCount(discussionItemList.size());

            searchView = findViewById(R.id.searchViewSearch);
```

```
            searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
                @Override
                public boolean onQueryTextSubmit(String query) {
                    return false;
                }

                @Override
                public boolean onQueryTextChange(String newText) {
                    List<DiscussionItem> filteredList = new
ArrayList<>();

                    for (DiscussionItem item : discussionItemList) {
                        if
(item.getTopic().toLowerCase().contains(newText.toLowerCase())) {
                            filteredList.add(item);
                        }
                    }

                    adapter.updateList(filteredList);
                    return true;
                }

            });
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            CharSequence text = "Fail to get data!";
            int duration = Toast.LENGTH_SHORT;
            Toast toast =
Toast.makeText(com.example.forum.students.StudViewForum.this, text,
duration);
            toast.show();
        }

    });
}
```

5.0 Testing Strategies, Approaches and Results

5.1 Inspections

During the inspection phase, our team engaged in a review of the codebase. This involved a personal testing approach, where each team member revisited their own code to ensure the logic was sound and all functionalities operated as intended.

The process began with a comprehensive code read-through. Team members carefully examined their code, scrutinizing each line to confirm that the logic and flow aligned with the intended functionality. This step aimed to identify any discrepancies between the written code and the original design specifications.

Subsequently, functional testing was conducted. Various scenarios were executed within the code to validate expected outcomes. Team members confirmed that conditional statements and loops operated as intended, and they checked for edge cases while handling unexpected inputs gracefully.

Additionally, a focus was placed on error handling. The implementation of error-handling mechanisms was verified, and the team checked how the code responded to unexpected situations. Attention was given to ensuring that appropriate error messages were displayed in case of unforeseen events.

Results from this personal testing approach served as a final checkpoint before the code integration phase. It helped identify and rectify any potential oversights or logical errors, contributing to the overall code quality. This individualized code reevaluation can effectively reinforce individual accountability and enhance the robustness of our application by minimizing the likelihood of logical errors.

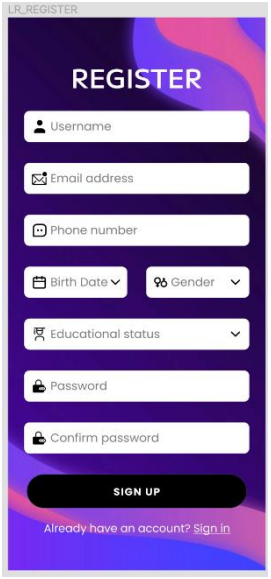
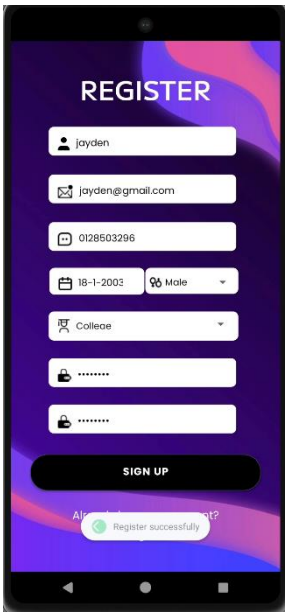
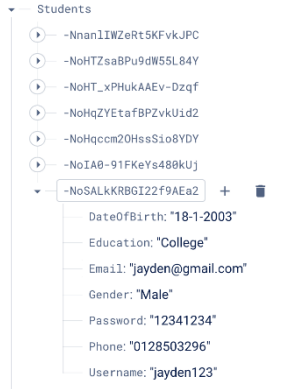
5.2 Unit Testing (Manual Testing)

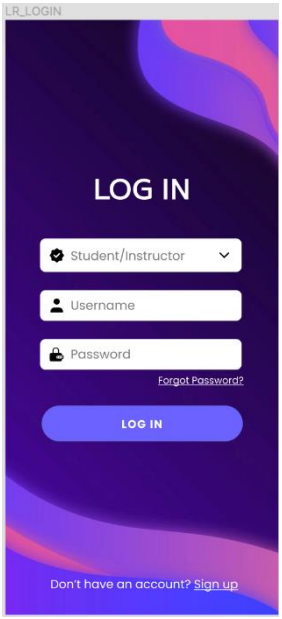
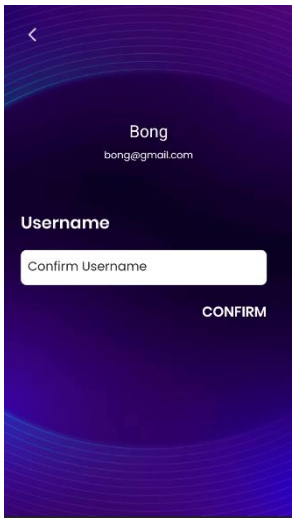
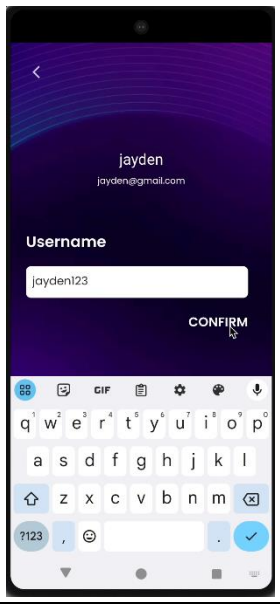
Unit testing, a crucial component of our quality assurance process, involves thorough examination of each module and function within the EduEmpower application. This form of testing simulates the perspective of the end user, ensuring that every aspect works as expected and behaves predictably.

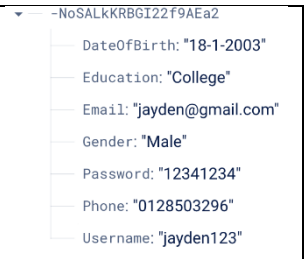
Simulation of End User Perspective:
During unit testing, the process involves replicating the end user's interaction with each module. This ensures that functionalities work seamlessly and align with user expectations. The objective is to provide a testing environment that closely mirrors the end user's experience, allowing for the identification of potential issues related to user interactions and expectations.

Verification of Expected Outcomes:
The purpose of unit testing is to verify that each module and function produces the anticipated results. It aims to ensure that the behavior remains consistent and predictable under various scenarios. This objective emphasizes the importance of validating that the software functions as intended, meeting the expected outcomes across different conditions and inputs.

5.2.1 Account Module

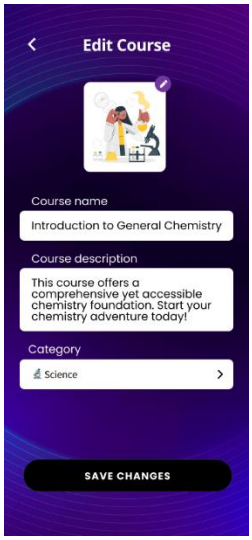
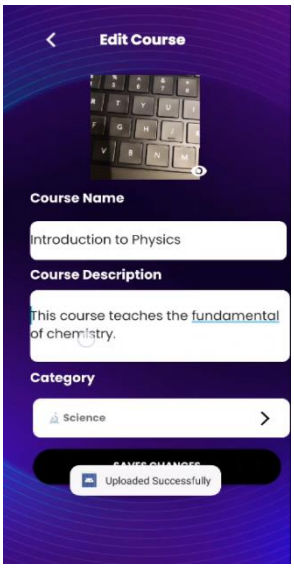
Test Case Description	Expected Results	Actual Result	Remarks
Register and save data in Firebase	Users should be able to register their account and save their profile details in the Firebase database successfully for the next time login without the need to register new account again. 	Users successfully register their account and save their filled personal details in the Firebase database.  	Test case has passed
Login successfully	For users who have done their registration, they should be able to login into their	Registered users can login into their own account successfully with their own email	Test case has passed.

	account seamlessly with the registered email and password. 	address and password. 	
Update successfully	Users should be able to update their personal details when they want to, and the updated details should be reflected in the Firebase database. 	Users can update their details successfully and the updated version is reflected automatically and immediately in the Firebase Database. 	Test case has passed.

		 -NoSALkKRBGI22f9AEa2 DateOfBirth: "18-1-2003" Education: "College" Email: "jayden@gmail.com" Gender: "Male" Password: "12341234" Phone: "0128503296" Username: "jayden123"	

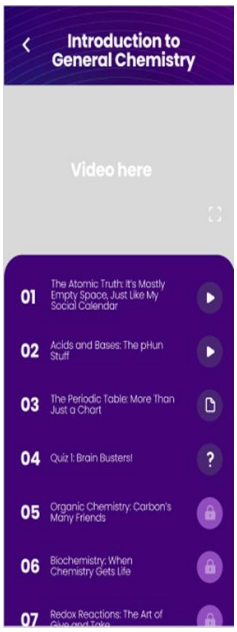
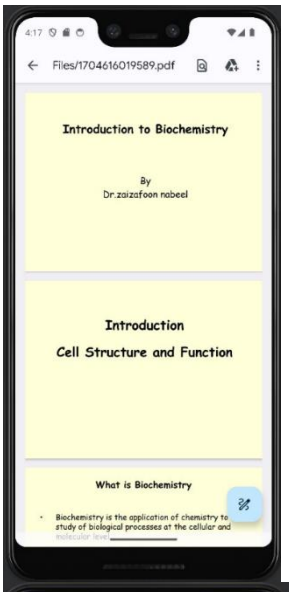
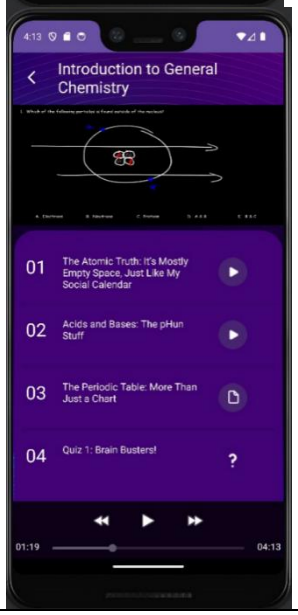
5.2.2 Course Module

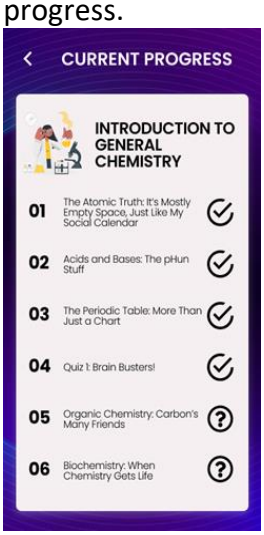
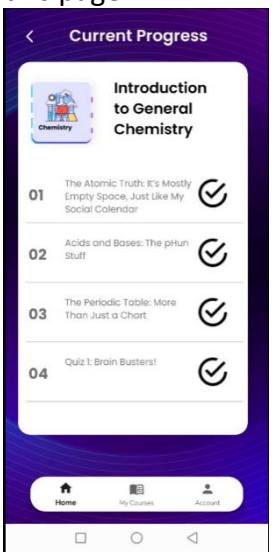
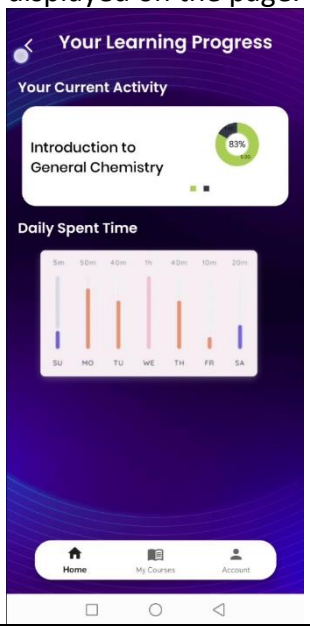
Test Case Description	Expected Results	Actual Result	Remarks
Add Course	Instructors should be able to add courses by clicking the add course button. After instructors successfully add courses, they should be able to see the course they have added in that interface. 	Although the design is slightly different, instructors can add courses by clicking the add course button. It shows that instructors can successfully retrieve the course from database and shown in that interface. 	Test case has passed.
Edit Course	Instructors should be able to edit the course that was added last time. For example, they can edit the image and text.	Instructors can edit the course added last time. They successfully upload the image and edit the course information when	Test case has passed.

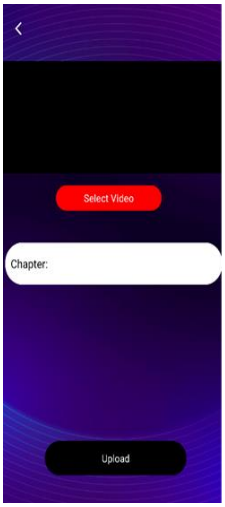
		clicking the saves changes button. Finally, the updated information has been successfully retrieved from the database and has been updated everywhere as well as the student view course page. 	
Browse Course	Students should be able to browse the variety of courses prepared by instructors.	Students can browse the variety of courses prepared by instructors.	Test case has passed.

	 My Courses Search a course Discover courses WEB DEVELOPMENT Web development is the work involved in developing a website for the Internet World Wide Web or an intranet (a private network). Ongoing courses See All Accounting 20 Courses Biology 15 Courses Completed courses See All INTRODUCTION TO GENERAL CHEMISTRY Explore the fundamental principles of chemistry in this introductory course. Retake Home My Courses Account	 My Course Search courses Discover courses Ongoing courses SEE ALL Accounting 20 courses Biology 15 courses Completed courses SEE ALL Introduction to General Chemistry RETAKE Home My Courses Account	

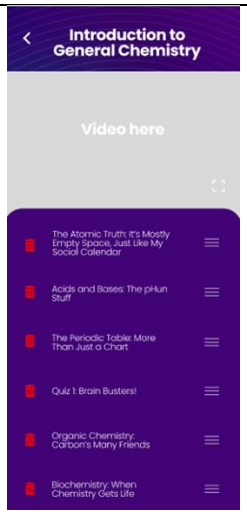
5.2.3 Chapter Module

Test Case Description	Expected Results	Actual Result	Remarks
Study and manage class material	Students should be able browse through the chapters and choose one to study. We expect that students can play the video or read the file after they clicked the button. 	Students can browse through the chapters and choose any chapter to study on. Besides, students can also play the video and read the file after they clicked the button.  	Test case has passed.

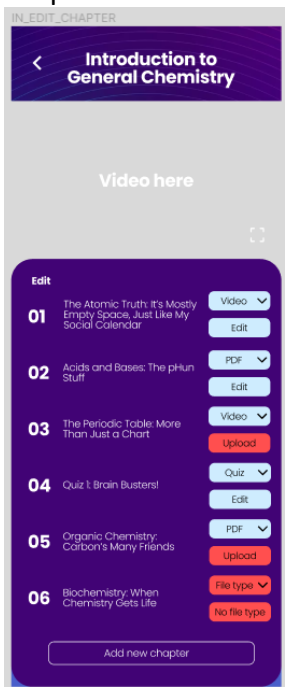
Progress tracker	Students should be able to view their current progress.  Students should be able to view a few aspects such as a pie chart showing the current progress, daily spent time and achievements.  	Students can view their current when navigate to this page.  Students can view aspects such as a pie chart showing the current progress and daily spent time. However, achievements are not displayed on the page.  	Test case has passed.
Manage class material	Instructor should be able to add a new chapter by uploading a	Instructor can add a new chapter by uploading a	Test case has passed.

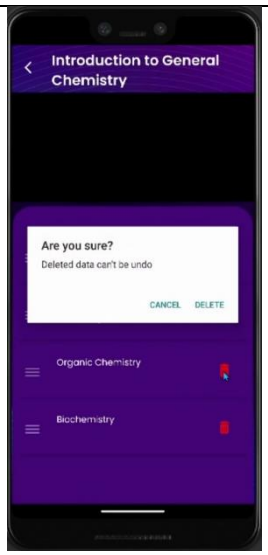
	video file and inserting a new chapter name. Then, the new chapter should be inserted into the firebase. 	video file and inserting a new chapter name.  A message will pop up once the new chapter is inserted into the firebase. 	

Manage class material	Instructor should be able to a add new chapter by uploading a pdf file and inserting a new chapter name. Then, the new chapter should be inserted into the firebase. 	Instructor can add new chapter by uploading a pdf file and inserting a new chapter name.  A message will pop up once the new chapter is inserted into the firebase. 	Test case has passed.
Manage class material	Instructor should be able to delete a specific chapter by clicking the delete button beside the specific chapter name. Then, the new chapter should be removed from the firebase.	Instructor can delete a specific chapter by clicking the delete button beside the specific chapter name. Then, the new chapter should be removed from the firebase.	Test case has passed.

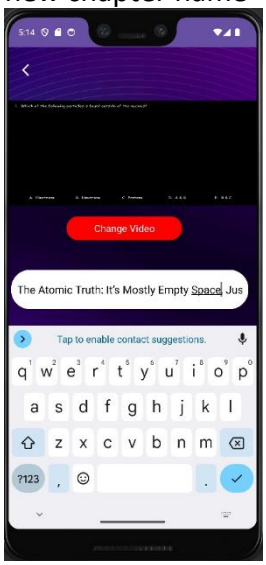


Instructor should be able to edit the chapter





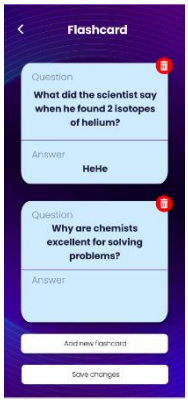
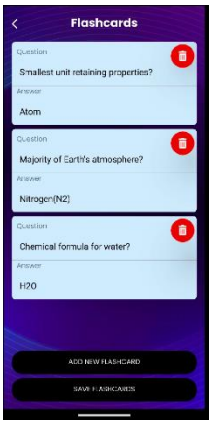
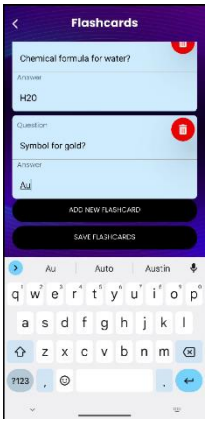
Instructor can edit the chapter by uploading the new video or changing the new chapter name



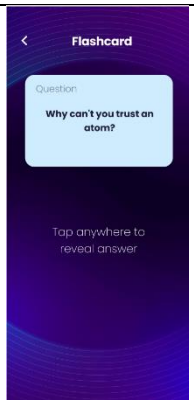
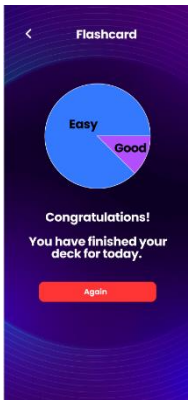
[pdf file type]

		 5:15 1/11 Select File The Periodic Table: More Than Just a Chart Save Changes	

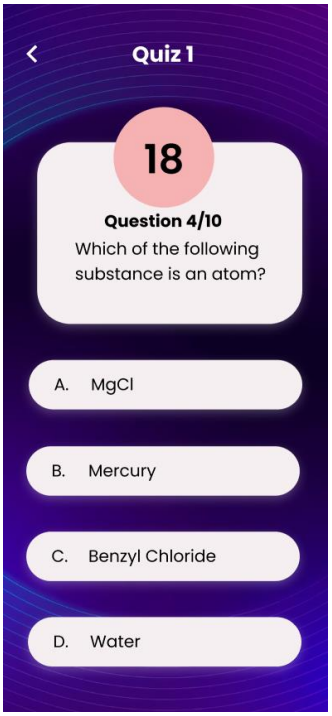
5.2.4 Flashcard Module

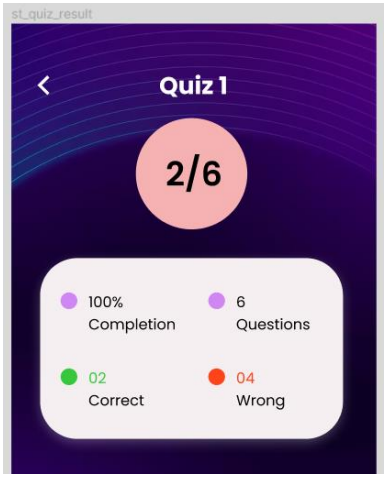
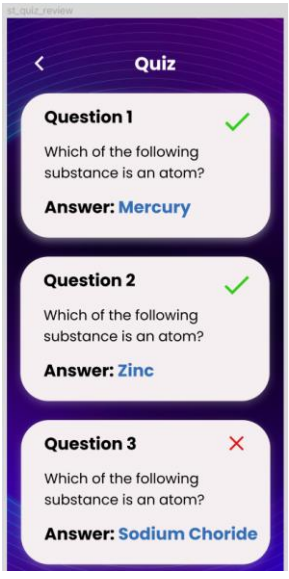
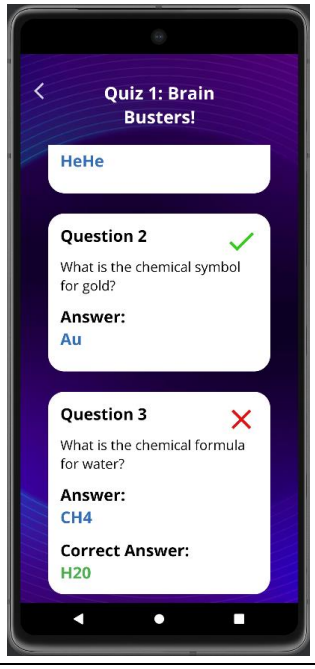
Test Case Description	Expected Results	Actual Result	Remarks
Manage flashcard	Instructors should be able to manage the flashcard questions by filling the question and correct answer. When the delete button is pressed, it should be able to delete the existing flashcards. When the add flashcards button is pressed, it should be able to add another flashcard for input, and save changes should be able to store the question to our database. 	Instructors can fill in the question and correct answer, delete the existing flashcards, after pressing the add flashcard button, new flashcard is added, and changes are shown to be updated successfully upon pressing save changes button.   	Test case has passed.
Study & browse flashcard	Students should be able to study the flashcard efficiently. Upon entering the flashcard feature, students should be	Students are successfully able to retrieve questions stored in the database and proceed to study the flashcard deck. The	Test case has passed.

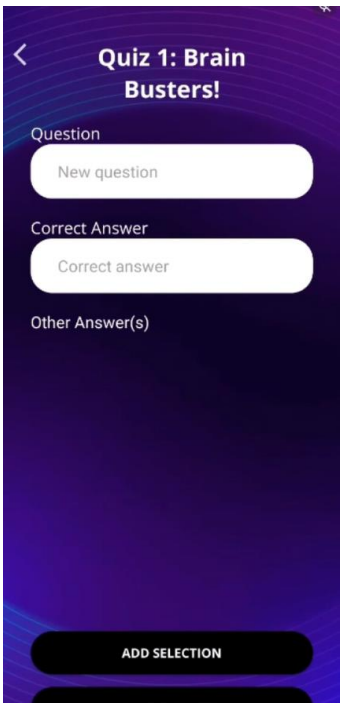
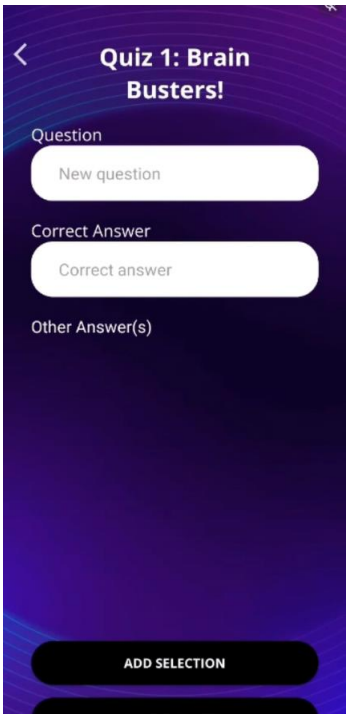
	presented with questions prepared by the instructor, retrieved from the database. Students should be able to answer each question in their mind and review the answer for each question by tapping on the screen.  After showing the answer, students should be able to rate their performance for each flashcard.  After the rating, students should be able to proceed to the next flashcard until completion.	system displays question of the flashcard at first and students are able to view the answer by tapping on 'show answer' button.  Students are successfully able to rate their performance for current flashcard after viewing the answer.  After the rating, students are able to move on to the next flashcard until finishing the deck.	

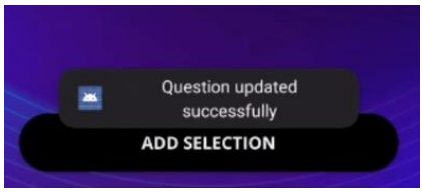
			
View performance	After completed the revision, students should be able to view the overall statistics of performance on studying the flashcards based on their rating. Also, students should be able to choose to have the revision again. 	Application accurately displayed the statistics, specifically a pie chart of the overall performance based on the rating to the students' understanding. Students can also repeat the flashcard revision by tapping the again button. 	Test case has passed.

5.2.5 Quiz Module

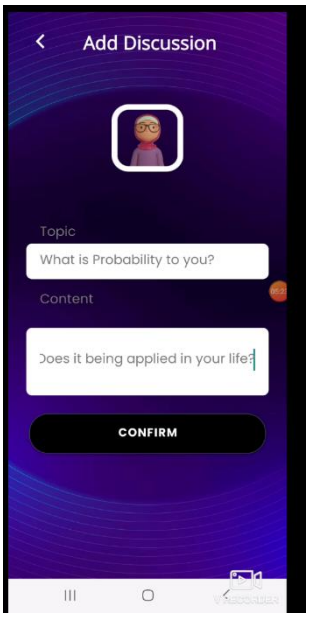
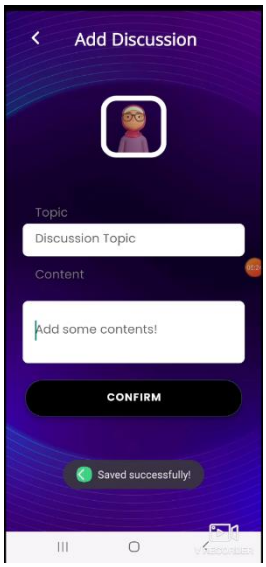
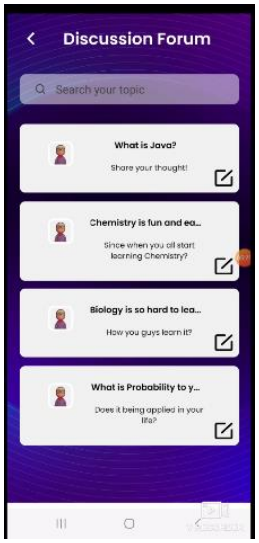
Test Case Description	Expected Results	Actual Result	Remarks
Attempt quiz	Students should be able to attempt the quiz seamlessly. Upon attempting the quiz, students should be presented with questions prepared by the instructor, retrieved from the database. Students should be able to proceed through the quiz, answering each question one after another until the quiz concludes. 	Students are successfully able to retrieve questions stored in the database and proceed to attempt each question. The system displays questions as intended, and students can seamlessly navigate through the quiz. The countdown timer correctly resets every time they move on to the next question. 	Test case has passed.
View results	In a six-question test scenario, we will input two correct answers and four incorrect answers. The expected outcome should reflect the results described above.	Application accurately display the information of the quiz result.	Test case has passed.

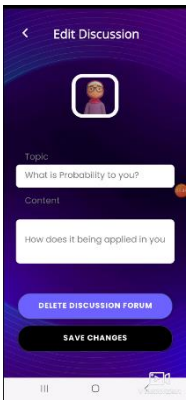
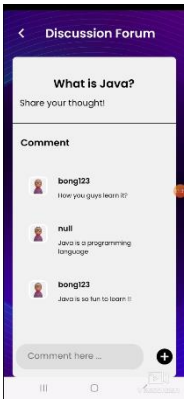
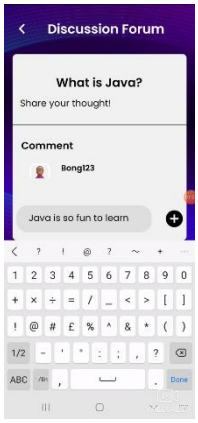
	Students should be able to view the results of the attempted quiz, showing the information of the quiz like number of correct and wrong answers, as well as the completion percentage. 		
Review answers	Students should be able to review the attempted questions and answers of the quiz. 	Students can view the attempted questions and answers, with the additional information of the attempted false answer, and the provided correct answer for reference. 	Test case has passed.

Manage quiz questions	Instructors should be able to manage the quiz questions by filling the question and correct answer box, as well as adding a selection to the question. When add selection button is pressed, it should be able to add another white blank box for input, and save changes should be able to store the question to our database. 	Instructors can fill in the question and correct answer box, after pressing the add selection button, new box is added, and question are shown to be updated successfully upon pressing save changes button. 	Test case has passed.

		 Quiz 1: Brain Busters! Question quesitoon Correct Answer answer Other Answer(s) Answer 1 Answer 2 Answer 3 ADD SELECTION  Question updated successfully ADD SELECTION	

5.2.6 Discussion Module

Test Case Description	Expected Results	Actual Result	Remarks
Create discussion forum	Instructors should be able to create a discussion forum by adding the topic and relevant content of the discussion. Once pressing the confirm button, the topic and content should be uploaded into firebase successfully, and a pop-up message will be shown. 	Instructors can add a new discussion forum by adding the topic and content of it. The newly added discussion forum will be displayed in the view forum page.  	Test case has passed.
Edit discussion	Instructors should be able to edit the discussion topic and	Instructors can delete the discussion item and view the latest discussion items.	Test case has passed.

	content and delete the discussion. 		
Add Comment	After choosing a discussion item, users should have the capability to add comments, and these comments will be successfully stored in Firebase. 	Users' comments are added and displayed in recycle view successfully.  	Test case has passed.

5.3 Walk-throughs

In the code walk-through session that took place in the end of every sprint, the individual responsible for a specific code segment presents their code line by line to the internal development team, comprising all six team members. This collaborative and interactive process serves multiple purposes:

1. **Knowledge Sharing:**

   The person responsible for the code shares insights into their implementation choices, design rationale, and any noteworthy considerations. This fosters knowledge sharing within the team, ensuring that everyone gains a deeper understanding of the codebase.

2. **Comprehensive Understanding:**

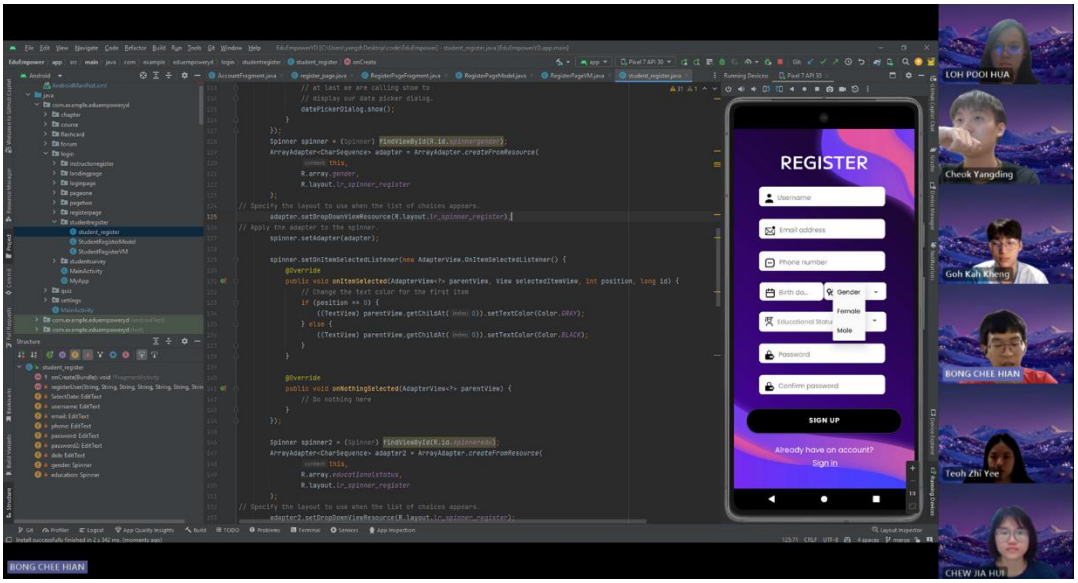
   The line-by-line presentation allows team members to gain a comprehensive understanding of the code's logic and functionality. It provides an opportunity for clarification on intricate details, ensuring that every team member is on the same page regarding the code being presented.

3. **Identifying Issues:**

   The walk-through serves as an effective forum for identifying potential issues or improvements. Team members can ask questions, offer suggestions, and collectively address any concerns related to code quality, performance, or adherence to coding standards.

4. **Ensuring Consistency:**

   By presenting the code to the entire team, consistency in coding standards and practices is reinforced. Any deviations from established conventions can be discussed and addressed, promoting a unified approach to development.



Process:

- **Presenter's Role:**

The individual responsible for the code takes on the role of the presenter. They go through their code line by line, providing explanations, rationale, and addressing any potential questions from the team.

- **Interactive Discussion:**

Team members actively participate in an interactive discussion. They can seek clarifications, suggest improvements, and discuss alternative approaches to certain code segments.

- **Documentation of Insights:**

Insights gained during the walk-through, including potential improvements or issues identified, are documented. This documentation serves as a valuable resource for future reference and continuous improvement.

- **Iterative Process:**

Code walk-throughs are conducted iteratively in the end of every sprint, ensuring that every team member has the opportunity to present their code and receive feedback. This iterative process contributes to a dynamic and collaborative development environment.

5.4 User Testing

To ensure the EduEmpower app aligns with both user expectations and functional requirements, a comprehensive interview survey was conducted. The participant, Yee Song Jun, excited to explore EduEmpower, engaged in a dialogue that unveiled valuable feedback.

- **Discovery of Courses:**
  The "Discover Courses" feature was highlighted as an accessible entry point, providing the user with a diverse range of subjects. The categorization of courses received positive feedback, indicating effective navigation.

- **View Learning Progress:**
  The "View Learning Progress" feature proved to be intuitive, offering a clear overview of completed courses and areas for improvement. The visual representation was specifically praised for aiding quick comprehension.

- **Course Exploration:**
  Upon selecting a course, the user found the organization of chapters seamless. The inclusion of various learning materials like videos, PDFs, and quizzes within each chapter received positive recognition.

- **Quiz Attempt Experience:**
  Navigating the quiz section was described as a smooth process. The user-friendly layout was acknowledged for enhancing focus during the quiz attempt and subsequent review.

- **Interactive Features - Flashcards and Discussion Forums:**
  Both flashcards and discussion forums were well-received. Flashcards were lauded as an effective interactive study tool, and the discussion forums added a collaborative aspect to the learning experience.

- **User Settings - Update User Details:**
  The user settings section, particularly the "Update User Details" feature, was described as straightforward and easy to use. Updating personal details was perceived as an intuitive process.

- **Suggestions for Enhancement:**

While the overall experience was positive, the user suggested the incorporation of a feature for personalized course recommendations based on individual interests. This insightful feedback opens avenues for future improvements.

The user testing interview survey provided invaluable insights into the user experience of EduEmpower. Positive feedback on usability, features, and navigation was reinforced, and the constructive suggestion for personalized recommendations highlights areas for potential enhancement. The findings from this survey, along with the consent letter and full interview script, can be found in the appendix.

5.5 Non-functional requirements testing

In our pursuit of ensuring the optimal performance and security of the EduEmpower mobile application, we conducted rigorous testing to validate the specified non-functional requirements. The focus of our non-functional requirements centers around response time and security.

1. Response Time Assessment:

Due to the challenges we face in direct testing with a large user pool, a research-driven approach was adopted to evaluate the application's response time. Firebase, renowned for its stable server infrastructure capable of accommodating a substantial number of users, was identified as a strategic choice. The assessment involved scaling up server interactions to gauge the application's response time under varying loads.

Leveraging Firebase's robust server capabilities, we emulated scenarios simulating increased user engagement, akin to a surge in application usage. The findings demonstrated consistent and efficient response times, affirming the stability of Firebase's server infrastructure. This research-driven approach, coupled with Firebase's reputation, provided valuable insights into the application's responsiveness, ensuring a user-friendly experience even in conditions mimicking peak usage.

2. Security Testing:

To fortify the security of EduEmpower, stringent measures were implemented during the login process. We ensured that only users with matched passwords could successfully log in, preventing unauthorized access. This approach adds an additional layer of protection to user accounts, reducing the risk of breaches through brute force or password-guessing attacks.

Moreover, we rigorously tested the application to guarantee that user-specific information remains confidential and is not revealed in unintended contexts. This includes stringent access controls and encryption practices to safeguard user data. Our security testing procedures encompassed vulnerability assessments, penetration testing, and exhaustive code reviews to identify and rectify potential vulnerabilities.

6.0 Work Completed & Contributions

6.1 Work Allocation

Role and Responsibilities

Name	Role	Responsibility
Cheok Yangding	Project Manager	• Tasked with problem statement, project goals and scope • Assign role and responsibilities to team member • Ensure report is delivered on time
Bong Chee Hian	Analyst	• Tasked with survey and findings, • data analysis from responses received • doing explanation based on the statistics collected
Chew Jia Hui	Editor / Reviewer	• Tasked with modules, functional and non-functional requirements • Review report's content for coherence and logical flow • Edit for grammar, punctuation, and spelling errors
Teoh Zhi Yee	Editor / Reviewer	• Tasked with modules, functional and non-functional requirements • Check for factual accuracy and consistency • Ensure report follows the format
Goh Kah Kheng	Analyst	• Tasked with survey and findings (Google Form) • Prepare Google Form for data collection
Loh Pooi Hua	Researcher	• Tasked with introduction and project objectives • Collect relevant data from various sources • Provide accurate and up-to-date information

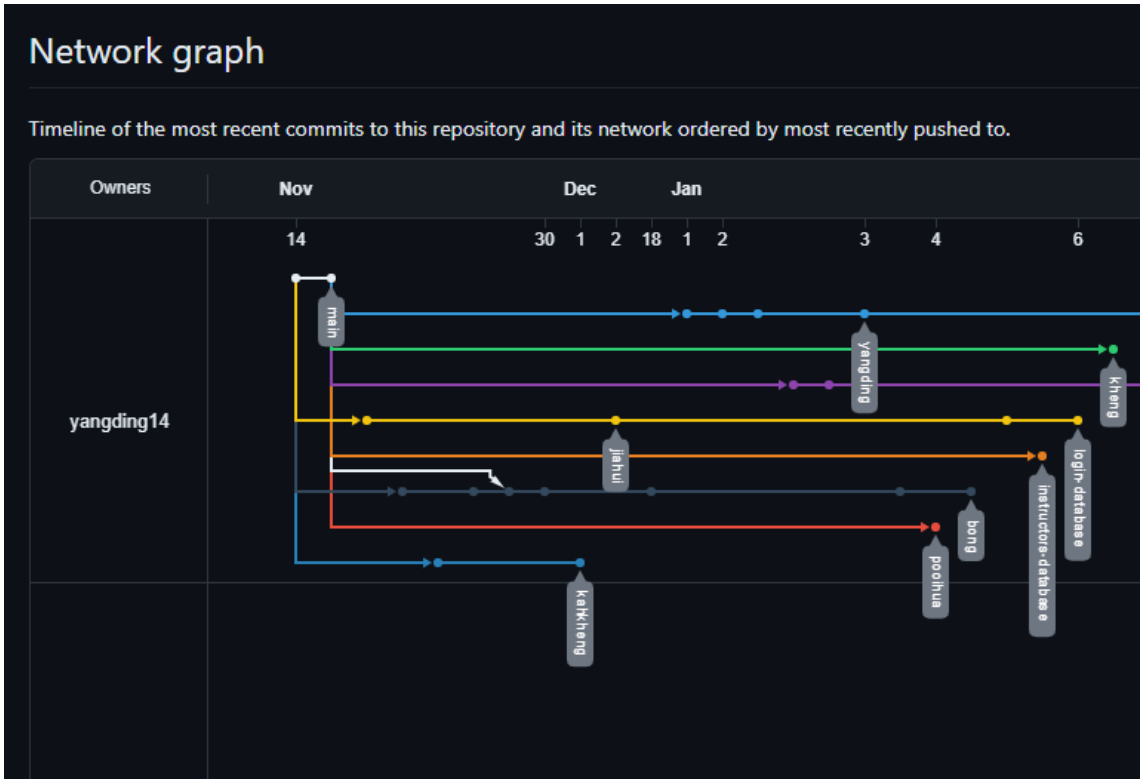
Modules Completed

Person-In-Charge	Module (Epic)	Functional requirement
Goh Kah Kheng	A: Class Material Module	Study and manage class material
		Progress tracker
		Manage class material
Loh Pooi Hua	B: Course Module	Browse courses
		Manage courses
		Course recommendation

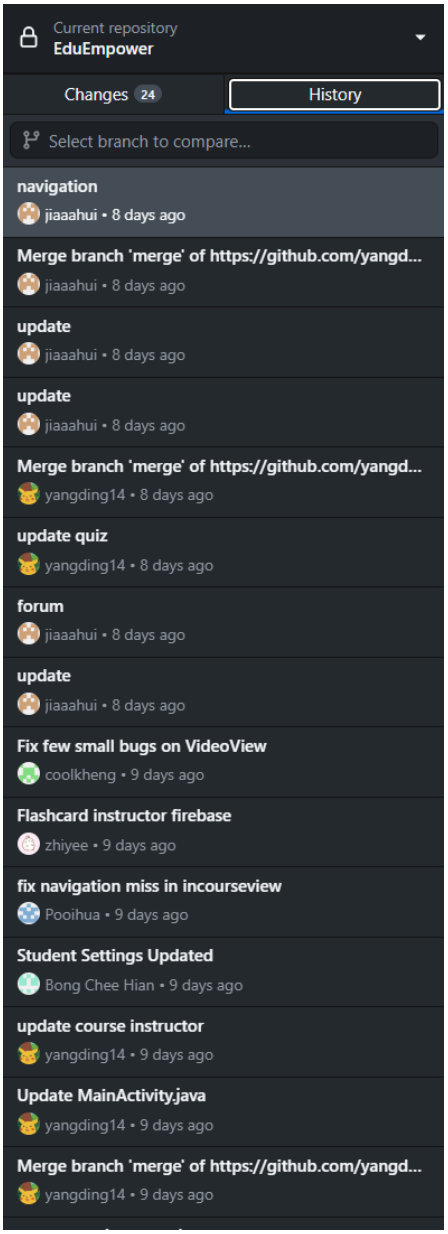
Teoh Zhi Yee	C: Flashcard Module	Browse and manage flashcards
		Flashcard recommendation
		Reward-based system for flashcards
Cheok Yangding	D: Quiz module	Attempt quiz and review quiz
		Manage quiz
		Analyze quiz
Chew Jia Hui	E: Discussion Module	Create discussion forum
		Add comment
		Edit/delete comment
Bong Chee Hian	F: Account Module	Log in and register account
		User settings
		Customer service/FAQ

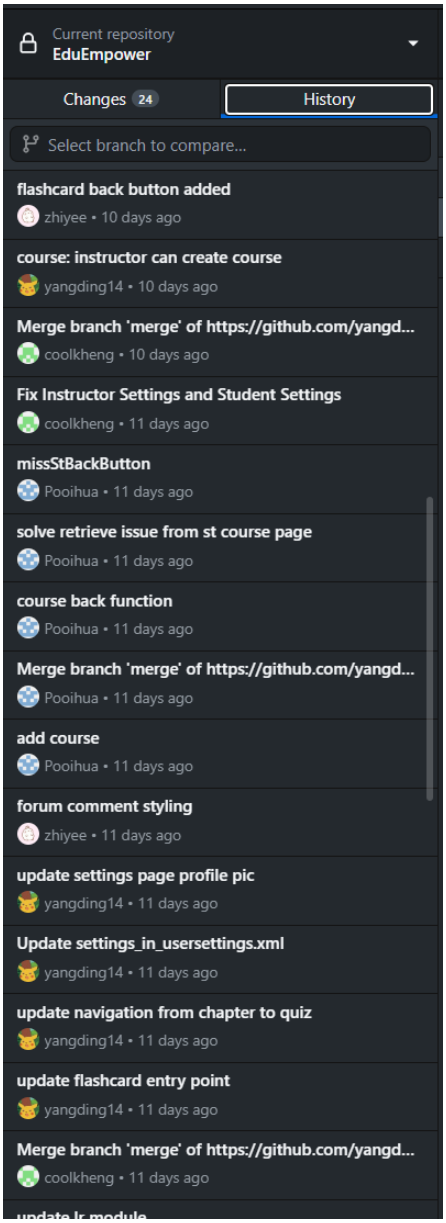
6.2 Git Commit Log

In the initial phase, each team member started individual feature development by pushing code to their respective branches, as depicted in the commit log network. Each branch housed distinct code segments labelled with either the contributor's name or the specific feature under development. Following this, code reviews and testing were conducted internally, and iterative improvements were implemented.



As the project progressed, challenges arose due to the accumulation of large files within individual branches. Recognizing the complexities associated with utilizing the Git merge feature, a decision was made to simplify the process. All Java files, resource files, AndroidManifest, and Gradle files were manually consolidated into a single main project. Subsequently, this unified project was pushed to a designated 'merge' branch, serving as the default branch for all team members. This branch became the focal point for ongoing work, streamlining collaboration and facilitating a cohesive development environment. Each team member contributed to this unified codebase, ensuring a more efficient and manageable project structure. The commit log for this 'merge' branch reflects the collective efforts and progress made by the entire team.


Current repository
EduEmpower
Changes 24
History
Select branch to compare...
navigation
jiaaahui • 8 days ago
Merge branch 'merge' of https://github.com/yangd...
jiaaahui • 8 days ago
update
jiaaahui • 8 days ago
update
jiaaahui • 8 days ago
Merge branch 'merge' of https://github.com/yangd...
yangding14 • 8 days ago
update quiz
yangding14 • 8 days ago
forum
jiaaahui • 8 days ago
update
jiaaahui • 8 days ago
Fix few small bugs on VideoView
coolkheng • 9 days ago
Flashcard instructor firebase
zhiyee • 9 days ago
fix navigation miss in incourseview
Pooihua • 9 days ago
Student Settings Updated
Bong Chee Hian • 9 days ago
update course instructor
yangding14 • 9 days ago
Update MainActivity.java
yangding14 • 9 days ago
Merge branch 'merge' of https://github.com/yangd...
yangding14 • 9 days ago


Current repository
EduEmpower
Changes 24
History
Select branch to compare...
flashcard back button added
zhiyee • 10 days ago
course: instructor can create course
yangding14 • 10 days ago
Merge branch 'merge' of https://github.com/yangd...
coolkheng • 10 days ago
Fix Instructor Settings and Student Settings
coolkheng • 11 days ago
missStBackButton
Pooihua • 11 days ago
solve retrieve issue from st course page
Pooihua • 11 days ago
course back function
Pooihua • 11 days ago
Merge branch 'merge' of https://github.com/yangd...
Pooihua • 11 days ago
add course
Pooihua • 11 days ago
forum comment styling
zhiyee • 11 days ago
update settings page profile pic
yangding14 • 11 days ago
Update settings_in_usersettings.xml
yangding14 • 11 days ago
update navigation from chapter to quiz
yangding14 • 11 days ago
update flashcard entry point
yangding14 • 11 days ago
Merge branch 'merge' of https://github.com/yangd...
coolkheng • 11 days ago

7.0 Appendix

Interview Consent Form

Consent for Participation in User Testing Interview Survey

I, Yee Song Jun, hereby provide my voluntary consent to participate in the user testing interview survey for the EduEmpower mobile application. I understand that the purpose of this survey is to gather insights into my user experience, preferences, and feedback regarding the functionality and usability of EduEmpower.

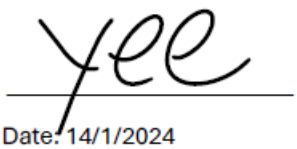
I acknowledge that my participation in this study is entirely voluntary, and I reserve the right to withdraw at any point during the interview without facing any consequences. My decision to participate or withdraw will not impact my relationship with the EduEmpower team or any associated parties.

I understand that all information collected during the interview will be treated confidentially. My name and any personally identifiable information will be kept confidential and will not be disclosed without my explicit consent. The data gathered will only be used for the purpose of improving the EduEmpower app.

I am aware that the interview may be recorded for accuracy in capturing my feedback. I understand that the recordings will be securely stored and will only be accessible to the research team. These recordings will be deleted after the completion of the study.

I have read and understood the information provided above, and I willingly agree to participate in the user testing interview for the EduEmpower app.

Participant's Signature:



Date: 14/1/2024

Interview Script

INTERVIEW SCRIPT
USER TESTING

Interviewer: Hello and thank you for participating in our user testing session today. Before we dive into the app, let me provide you with a brief overview of EduEmpower. It's a comprehensive educational platform with a variety of courses and interactive learning materials. Our primary goal is to create a seamless and enriching learning experience tailored to users like yourself. Today, our focus is to understand how well the app aligns with its functional requirements. Now, let's get started!

Interviewee: Thank you for having me! I'm excited to see what EduEmpower has to offer.

Interviewer: Fantastic! As you navigate the app, our first task is to discover courses. Can you walk me through how you would go about finding a course of interest?

Interviewee: Certainly! I noticed the "Discover Courses" option on the home page, so I'd naturally click on that to explore what subjects are available.

Interviewer: As you explore, do you find the course offerings diverse and easily accessible?

Interviewee: Yes. There's a wide variety of subjects, and the layout makes it easy to quickly see what's available. The categorization is particularly helpful.

Interviewer: That's wonderful feedback. Now, let's move on to the "View Learning Progress" feature. How intuitive is it for you to track your academic achievements and identify areas for improvement?

Interviewee: I found it quite intuitive. The "View Learning Progress" feature provides a clear overview of completed courses and areas where I might need to focus more. The visual representation makes it easy to grasp.

Interviewer: Thank you. As we dive into specific courses, imagine you've selected a course. How do you find the chapters and learning materials within it?

Interviewee: When I enter a course, I see chapters neatly organized. The navigation is seamless, and each chapter has a variety of learning materials like videos, PDFs, and quizzes.

Interviewer: Now, let's say you want to attempt a quiz within a chapter. Can you walk me through that experience?

Interviewee: Uh, I'd click on the quiz, answer the questions, and then review my answers afterward. The process is smooth, and the user-friendly layout makes it easy to focus on the questions.

Interviewer: Beyond quizzes, we also have interactive flashcards and discussion forums. How do you find these features?

Interviewee: The flashcards are a great interactive study tool, and the discussion forums add a collaborative aspect to learning. I can add topics, join discussions, and comment on various subjects.

Interviewer: It's wonderful to hear that you're finding these features valuable. Lastly, in the user settings section, where you can update your details, is the process intuitive for you?

Interviewee: Yes, updating my details is straightforward. The "Update User Details" feature is easy to locate and use.

Interviewer: Thank you for your detailed feedback. It's clear you've had a positive experience so far. Is there anything you think could be improved or any additional features you'd like to see?

Interviewee: Overall, it's been a positive experience. Perhaps adding a feature for personalized course recommendations based on my interests could enhance the experience.

Interviewer: That's a valuable suggestion, and we appreciate it. Your feedback is immensely helpful as we continue to refine EduEmpower. Thank you for your time and insights today!

Interviewee: Happy to help! It's been a pleasure exploring the app.