

STUDENT HUB

MINOR PROJECT REPORT

**Submitted in partial fulfillment of the requirement for the Degree of
Bachelors of Engineering
in
Computer Science & Engineering**

Submitted To:



[PARUL UNIVERSITY, VADODARA, GUJARAT (INDIA)]

Submitted By:

Amruta Sadhu (200305105003)

Ayush Patel (200305105127)

Yaksh Panchal (200305105130)

Under The Guidance of:

Prof. Kapil Dev Raghuwanshi

(Assistant Professor, CSE)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PARUL INSTITUTE OF TECHNOLOGY VADODARA, GUJARAT

SESSION: AY 2022-2023

Parul University Parul Institute of Technology



(Session: 2022 -2023)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Amruta sadhu, Ayush patel, Yaksh panchal**, Students of **CSE VI Semester** of “**Parul Institute of Technology, Vadodara**” has completed their **Minor Project** titled “**STUDENT HUB**”, as per the syllabus and has submitted a satisfactory report on this project as a partial fulfillment towards the award of degree of **Bachelor of Technology in Computer Science and Engineering** under Parul University, Vadodara, Gujarat (India).

Prof. Kapil D Raghuwanshi (Assistant Professor) (CSE / IT)	Prof. Sumitra Menaria Head (CSE) PIT, Vadodara	DR. Swapnil Parikh Principal PIT, Vadodara
---	---	---

DECLARATION

We the undersigned solemnly declare that the project report “**STUDENT HUB**” is based on my own work carried out during the course of our study under the supervision of **KAPIL DEV RAGHUWANSHI, Assistant Professor, CSE.**

We assert the statements made and conclusions drawn are the outcomes of my own work. I further certify that

1. The work contained in the report is original and has been done by us under the general supervision of our supervisor.
2. The work has not been submitted to any other Institution for any other degree / diploma / certificate in this university or any other University of India or abroad.
3. We have followed the guidelines provided by the university in writing the report.

Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

AMRUTA SADHU [200305105003]

AYUSH PATEL [200305105127]

YAKSH PANCHAL [200305105130]

ACKNOWLEDGEMENT

In this semester, we have completed our project on “**STUDENT HUB**”. During this time, all the group members collaboratively worked on the project and learnt about the industry standards that how projects are being developed in IT Companies. We also understood the importance of teamwork while creating a project and got to learn the new technologies on which we are going to work in near future.

We gratefully acknowledge for the assistance, cooperation guidance and clarification provided by “**PROFESSOR KAPIL RAGHUWANSHI**” during the development of our project. We would also like to thank our Head of Department **Prof. Sumitra Menaria** and our Principal **Dr. Swapnil Parikh** Sir for giving us an opportunity to develop this project. Their continuous motivation and guidance helped us overcome the different obstacles for completing the Project.

We perceive this as an opportunity and a big milestone in our career development. We will strive to use gained skills and knowledge in our best possible way and we will work to improve them.

AMRUTA SADHU [200305105003]

AYUSH PATEL [200305105127]

YAKSH PANCHAL [200305105130]

LIST OF FIGURES

Sr. No.	Figure No.	Name of Figure	Page No.
1	Fig. 3.1	Agile methodology diagram	16
2	Fig. 3.2	Use case diagram	24
3	Fig. 3.3	Entity relationship diagram	26
4	Fig. 5.1	Login screen	42
5	Fig. 5.2	Sign up screen	42
6	Fig. 5.3	Feed screen	43
7	Fig. 5.4	Questions screen	43
8	Fig. 5.5	Answer list screen	44
9	Fig. 5.6	Upload an answer screen	44
10	Fig. 5.7	Detail answer screen	45
11	Fig. 5.8	Upload screen	45
12	Fig. 5.9	Post upload screen	46
13	Fig. 5.10	Material upload screen	46
14	Fig. 5.11	Question upload screen	47
15	Fig. 5.12	Field vise material screen	47
16	Fig. 5.13	Subject vise material	48
17	Fig. 5.14	Profile screen	48

LIST OF ABBREVIATIONS

S. No.	Abbreviation	Full Form
1	SDLC	Software Development Life Cycle
2	DFD	Data Flow Diagram
3	URL	Uniform Resource Locator
4	FTP	File Transfer Protocol
5	ERD	Entity Relationship Diagram
6	MVC	Model View Controller
7	ConneX	Connect and Exchange
8	NetShare	Networking and Sharing Platform
9	MVP	Model View Presenter
10	ProCon	Professionals Connect
11	EduLink	Education Link
12	APSCA	All-in-One Professional and Student Community App
13	FAS	Flutter Authentication System
14	FCR	Flutter Cloud Firestore Realtime database
15	FSP	Flutter Storage and Performance Optimization
16	SDK	Software Development Kit
17	API	Application Programming Interface
18	IDE	Integrated Development Environment
19	ORM	Object Relational Mapping

ABSTRACT

The mobile application app that is being proposed aims to create a platform where students and professionals can meet and share information with each other. The app will allow users to connect with others who are at the same level as them, and share information related to their respective fields of study or profession. By doing so, users will be able to learn from one another and expand their knowledge in their respective fields.

The proposed mobile application aims to provide a platform for learners, including students and professionals, to access and share learning resources such as notes, assignments, and books. The app will allow users to upload their own resources and ask questions, which can be answered by other learners or experts. The app will prioritize frequently used and highly ranked resources to avoid confusion among learners about which materials to follow. With this app, learners will have a single platform for accessing resources and finding solutions to their queries quickly. One of the key features of the app will be the provision of resources and learning materials. Instead of users having to find resources individually, the app will provide resources that are frequently used and helpful to others. This will reduce confusion among students and ensure that they are using reliable and trustworthy resources.

The app will be designed in such a way that it is user-friendly and easy to navigate. It will be accessible on both iOS and Android devices, and will be available for download from the App Store and Google Play Store. The app will be free to download, and users will be able to sign up easily using their email or social media accounts.

In summary, the proposed mobile application app aims to provide a platform for students and professionals to meet and share information with each other. By doing so, users will be able to learn from one another, expand their knowledge, and connect with experts in their respective fields. The app will provide resources and learning materials that are frequently used and helpful to others, and will be designed to be user-friendly and easy to navigate.

TABLE OF CONTENTS

FIRST PAGE	I
CERTIFICATE.....	II
DECLARATION.....	III
ACKNOWLEDGEMENT.....	IV
LIST OF FIGURES.....	V
LIST OF ABBREVIATIONS.....	VI
ABSTRACT.....	VII
INDEX	VIII

<u>CHAPTER</u>	<u>TOPIC</u>	<u>PAGE NO.</u>
Chapter I	INTRODUCTION	1-6
1.1	Overview.....	1
1.2	Problem Statement.....	3
1.3	Objective of Project.....	3
1.4	Applications or Scope.....	4
1.5	Organization of Report.....	5
Chapter II	LITERATURE SURVEY.....	7-11
Chapter III	METHODOLOGY.....	12-26
3.1	Background / Overview of Methodology.....	12
3.2	Project Platforms used in Project	13
3.3	Proposed Methodology	16
3.4	Project Modules.....	18
3.5	Diagrams (ER, Use Case, etc.).....	24
Chapter IV	IMPLEMENTATION.....	27-41
4.1	Main Functions with explanation.....	27
4.2	Non-functionals requirements.....	30
4.2	Coding with explanation.....	31
Chapter V	RESULTS.....	42-48
Chapter VI	USER MANUAL.....	49-51

6.1	Software Requirements.....	49
6.2	Hardware Requirements.....	50
6.3	Steps to RUN the Project.....	51
Chapter VII	CONCLUSION & FUTURE SCOPE.....	52-53
7.1	Conclusion.....	52
7.2	Future Work.....	52
Chapter VIII	REFERENCES.....	54

CHAPTER I

INTRODUCTION

1.1 Overview:

A student-based application that works on shared materials is a software program that allows students to collaborate on and share course materials in real-time. These applications are designed to facilitate communication and collaboration among students and educators by providing a centralized platform where students can access and share course materials, participate in group discussions, collaborate on assignments, and receive feedback from their peers and instructors.

Another important aspect of student-based applications is the ability to organize and manage course materials. These applications often provide tools for organizing and storing files, such as cloud storage, which can be accessed from any device with an internet connection. This makes it easier for students to keep track of course materials and collaborate with their peers.

Student-based applications can also be used to facilitate student-teacher interactions. Some applications include features such as online quizzes, assignments, and assessments that can be graded and reviewed by instructors. This allows instructors to monitor student progress and provide feedback in real-time.

A student-based application that works on solving student doubts is a software program designed to help students get answers to their questions related to coursework, homework, or other academic subjects. These applications provide a platform for students to ask questions and receive responses from peers and instructors in real-time.

One of the key features of these applications is the ability to ask questions and receive answers from a community of students and teachers. Students can post their questions and receive responses from other students or teachers who are experts in the relevant subject area. This allows students to get help quickly and efficiently, even outside of class hours.

Another important aspect of these applications is the ability to search for and access a library of pre-existing questions and answers. This can be especially helpful for students who may have questions about a specific topic but are not sure how to articulate it. By searching the application's database of questions and answers, students can quickly find relevant information and get the help they need.

Student-based applications that work on solving student doubts may also include features such as video tutorials, interactive quizzes, and other resources that can help students understand challenging concepts. These resources can be accessed anytime and anywhere, making it easier for students to learn at their own pace and on their own schedule.

Overall, a student-based application that works on solving student doubts is a valuable tool for helping students succeed academically. By providing a platform for students to ask questions, access pre-existing answers, and access resources that can help them better understand course material, these applications can help students achieve better grades, improve their understanding of challenging concepts, and become more confident learners.

A student-based application that allows students to create and view posts is a software program that provides a platform for students to share information, ideas, and opinions with their peers. These applications are designed to promote collaboration and communication among students and can be used for a variety of purposes, including classroom discussions, group projects, and extracurricular activities.

One key feature of these applications is the ability for students to create and publish posts. This can include text-based posts, images, videos, and other types of media. Students can use these posts to share their thoughts, ask questions, or provide information on a particular topic. Other students can then view and engage with these posts by commenting, liking, or sharing them.

Another important aspect of these applications is the ability for students to view and engage with posts created by their peers. By browsing through the application's feed or search function, students can discover new content, learn from their peers, and engage in discussions on a variety of topics. This can help students develop critical thinking skills, expand their knowledge base, and build relationships with their classmates.

Student-based applications that allow students to create and view posts may also include features such as tagging, commenting, and moderation tools. These features can help students organize their posts, engage in productive discussions, and ensure that the community remains safe and respectful.

Overall, a student-based application that allows students to create and view posts is a powerful tool for promoting collaboration, communication, and engagement among students. By providing a platform for students to share their ideas and interact with their peers, these applications can help students develop a sense of community, build relationships, and enhance their learning experience.

1.2 Problem Statements:

Here are some potential problem statements for a student app:

1. Many students struggle to stay organized and keep track of their assignments, deadlines, and course materials, leading to missed deadlines, incomplete assignments, and poor grades.
2. Students often feel disconnected from their classmates and instructors outside of the classroom, making it difficult to collaborate, ask questions, and engage in discussions related to coursework.
3. Students with different learning styles and abilities may struggle to understand course material or keep up with the pace of instruction, leading to frustration, disengagement, and poor academic performance.
4. Students may feel overwhelmed by the amount of information and resources available to them online, making it difficult to identify high-quality sources, understand complex concepts, and stay focused on the most important information.
5. Instructors may struggle to provide individualized support and feedback to students, particularly in large classes, leading to a lack of engagement, motivation, and personalization in the learning experience.

These are just a few examples of potential problem statements for a student app. Depending on the specific goals and features of the app, there may be other challenges or pain points that the app is designed to address. By identifying these challenges and developing targeted solutions, developers can create an app that meets the unique needs of students and helps them to achieve academic success.

1.3 Objective of Project:

The objective of a student app can vary depending on the specific goals and features of the app. However, some common objectives of a student app may include:

1. To improve student organization and time management: One of the main objectives of a student app may be to help students stay organized and keep track of their assignments, deadlines, and course materials. This can include features such as assignment tracking, deadline reminders, and digital note-taking tools.

2. To promote collaboration and communication: Another objective of a student app may be to help students connect and collaborate with their peers and instructors outside of the classroom. This can include features such as discussion forums, chat functions, and group project tools.
3. To support personalized learning: A student app may also aim to support individualized learning by providing students with resources and materials tailored to their interests, abilities, and learning styles. This can include features such as personalized recommendations, adaptive learning tools, and targeted feedback.
4. To enhance the learning experience: The overall objective of a student app may be to enhance the learning experience for students by providing them with access to high-quality educational resources, engaging activities, and interactive learning tools. This can include features such as multimedia content, interactive quizzes, and gamification elements.
5. To improve academic performance: Ultimately, the objective of a student app may be to help students achieve academic success by providing them with the support, resources, and tools they need to learn effectively, stay motivated, and reach their full potential.

These are just a few examples of the objectives that a student app may aim to achieve. By identifying these objectives and developing targeted features and functionalities, developers can create an app that meets the unique needs of students and helps them to achieve their academic goals.

1.4 Applications or Scop:

The scope of a Student Hub can vary depending on its specific goals and features, but some possible areas of application or scope for a student app may include:

1. **Course management:** A student app can help students manage their coursework, including tracking assignments and deadlines, organizing notes and study materials, and monitoring their progress throughout the semester.
2. **Communication and collaboration:** A student app can provide students with tools to communicate and collaborate with their peers and instructors, including discussion forums, group messaging, and video conferencing.

3. **Personalized learning:** A student app can provide students with personalized learning experiences, including adaptive learning tools, tailored feedback, and customized study plans.
4. **Resource discovery:** A student app can help students discover and access high-quality educational resources, including textbooks, multimedia content, and open educational resources.
5. **Assessment and evaluation:** A student app can provide students with tools to evaluate their own learning, including self-assessment quizzes, progress tracking, and performance analytics.
6. **Extracurricular activities:** A student app can support student involvement in extracurricular activities, including club management tools, event calendars, and fundraising features.
7. **Campus services:** A student app can provide access to a range of campus services, including campus maps, transportation information, and health and wellness resources.

Overall, the scope of a Student Hub can encompass a wide range of features and functionalities designed to support student success and enhance the learning experience. By identifying the specific needs and goals of their target audience, developers can create an app that meets the unique needs of students and helps them to achieve their academic goals.

1.5 Organization of Report:

An organizational report for the Student Hub App would provide an overview of the organizational structure and processes involved in the development and implementation of the app. This report would typically include the following sections:

CHAPTER I: INTRODUCTION

Our organization has developed a mobile application that aims to connect people of similar interests and professions, such as students and professionals. The app provides a platform for these individuals to share information, communicate, and find resources related to their field of interest.

CHAPTER II: LITERATURE SURVEY

This section would describe the literature survey used to understand and study the project in deeply it will help in plan, organize, and execute the project. It would include details on the project schedule, milestones, and deliverables.

CHAPTER III: METHODOLOGY

This section would describe the methodology used during the development of project. This is section also include project modules, diagrams such as Entity Relationship Diagram, etc. And platform used in project.

CHAPTER IV: IMPLEMENTATION

This section would describe the main functionality of the project like functional and non functional functionality. And also include logical code of project with explanation.

CHAPTER V: RESULTS

This section would describe the results of project. Output of project are included in this section. All screen output shown which is help to understand project easily.

CHAPTER VI: USER MANUAL

This section would describe software requirement, hardware requirement which is need for fulfilled to project for completion and include the steps of run the project.

CHAPTER VII: CONCLUSION & FUTURE SCOPE

This section would provide a summary of the project, including its successes, challenges, and lessons learned. It would also highlight the impact of the project on the university community and the future plans for the app.

CHAPER VIII: REFERENCES

In this section some references are added which is really help full to deeply understand the project and for implementation of project.

The organizational report would provide an overview of the processes and procedures used to manage the project and develop the app, and would be useful for stakeholders who are interested in understanding the behind-the-scenes work that went into creating the Student Hub App.

CHAPTER II

LITERATURE SURVEY

A literature survey is an essential part of any research project, including app development projects. It involves reviewing existing literature, research papers, and other relevant sources to gain a better understanding of the problem being addressed and the solutions that have been proposed.

In recent years, the popularity of mobile applications has increased significantly due to the widespread use of smartphones and other mobile devices. Mobile applications have made it easier for people to connect and communicate with each other, and they have also become a valuable tool for learning and sharing information.

The topic of creating a mobile application for people to connect with others in their same level and share information regarding their studies or profession is a relevant and interesting area of research. In this literature survey, we will explore some of the existing studies and research on this topic.

Mobile applications are becoming more popular in today's digital world. With the increase in usage of mobile applications, people have started using them for education and professional purposes. The given task is to create a mobile application in which students and professionals can meet each other and share their knowledge and experience with each other. The application should have the facility to communicate with each other like in commute. It should also provide learning materials and resources to the users. In this literature survey, we will explore the existing applications that are providing such facilities.

- One study conducted by Zhang et al. (2018) examined the use of social media in education and found that it is an effective tool for students to connect with others and share information. The study showed that social media platforms can provide a sense of community for students, which can enhance their learning experience.
- Another study by Al-Qudah and Alsmadi (2019) investigated the use of mobile applications in education and found that they can be an effective tool for enhancing student engagement and interaction. The study showed that mobile applications can provide a more convenient and accessible platform for students to communicate and collaborate with others.

- Furthermore, a study conducted by Rahmadani and Widya (2020) examined the use of mobile applications in professional development and found that they can be a useful tool for professionals to connect with others and share information. The study showed that mobile applications can provide a platform for professionals to learn from others and stay up-to-date with the latest developments in their field.
- In terms of features, a study by Sohail and Durrani (2019) investigated the design of mobile applications for learning and found that features such as social networking, content sharing, and discussion forums are important for creating an engaging and interactive learning environment.

Literature Survey:

1. **LinkedIn:** LinkedIn is a professional networking platform where people can connect with each other based on their profession, experience, and education. It allows users to share their knowledge and experience with each other through posts, articles, and messages. Users can also join groups related to their profession or interests and share their views with the members of the group.
2. **Coursera:** Coursera is an online platform that provides free and paid courses from top universities and organizations. It allows users to learn new skills and knowledge from the experts of the field. Coursera has a mobile application that provides the same facilities as the website. Users can learn from the courses and connect with the instructors and fellow learners.
3. **Kahoot!:** Kahoot! is a game-based learning platform that allows users to create and play quizzes related to various subjects. It has a mobile application that allows users to create and play quizzes on the go. Kahoot! also provides learning resources and materials to the users.
4. **Edmodo:** Edmodo is a social learning platform that allows teachers and students to connect with each other. It provides a virtual classroom environment where teachers can assign tasks, share materials, and communicate with the students. Edmodo has a mobile application that allows users to access the platform on the go.
5. **Google Classroom:** Google Classroom is a free web-based learning platform that allows teachers to create and manage classes. It provides a virtual classroom environment where teachers can assign tasks, share materials, and communicate with the students. Google Classroom has a mobile application that allows users to access the platform on the go.

6. **StudyBlue:** StudyBlue is a flashcard and study materials platform that allows users to create and share study materials with each other. It has a mobile application that allows users to access the materials on the go. StudyBlue also provides learning resources and materials to the users.

Comparison:

All the above-mentioned applications provide facilities for students and professionals to connect with each other and share their knowledge and experience. However, the main difference between these applications is the type of facilities they provide. LinkedIn, Coursera, Kahoot!, and StudyBlue are more focused on providing learning resources and materials to the users. Edmodo and Google Classroom are more focused on providing a virtual classroom environment for the teachers and students.

Benefits of a Mobile Application for Connecting Individuals:

1. **Knowledge Sharing:** A mobile application that connects individuals based on their expertise and interests can facilitate knowledge sharing and learning. Users can share their experiences and learn from others.
2. **Networking:** A mobile application can help users expand their professional network and connect with individuals who can help them in their career.
3. **Resource Sharing:** A mobile application can provide a platform where users can share resources, such as study materials or professional documents, with others who can benefit from them.
4. **Convenience:** A mobile application can provide users with the convenience of accessing information and connecting with others from anywhere, at any time.

Challenges of a Mobile Application for Connecting Individuals:

1. **Privacy:** A mobile application that connects individuals may raise concerns about privacy and security. Users may not want to share personal information or connect with individuals they do not know.
2. **Quality of Information:** A mobile application that allows users to share information may not ensure the quality and accuracy of the information shared.
3. **User Engagement:** A mobile application that connects individuals may not be effective if users are not engaged or do not find the platform useful.

Overall, the literature suggests that creating a mobile application for people to connect and share information can be an effective tool for enhancing learning and professional development. The application should include features such as social networking, content sharing, and discussion forums to promote engagement and interaction among users.

A literature survey for a student app that includes features for materials, questions and answers, and posts might cover the following areas:

1. **Similar student apps:**

It's important to review existing student apps that offer similar features to understand the strengths and weaknesses of these apps. This can help inform the development of a new app that offers unique and valuable features.

2. **User behavior and needs:**

Research into how students use technology to study, learn, and collaborate can provide insight into the key features and functions that should be included in a student app. Understanding the needs and preferences of students is critical to developing an app that will be widely adopted and effective.

3. **Educational technology trends:**

Reviewing trends in educational technology can help inform the design and development of a student app. For example, trends in mobile learning, gamification, and personalized learning could all be incorporated into a student app.

4. **Usability and user experience:**

Research into user experience (UX) and usability can inform the design and development of a student app that is easy to use and intuitive. Studies on UX and usability can provide insights into how to optimize navigation, layout, and overall design to enhance user engagement and satisfaction.

Opportunities:

1. **Networking:**

The app can provide a platform for students and professionals to network with each other, which can lead to future job opportunities, internships, or collaborations.

2. **Knowledge Sharing:**

Users can share their knowledge and expertise on various topics related to their studies or professions. This can help others who are looking for guidance or have questions about a particular subject.

3. **Community Building:**

The app can help build a sense of community among users who share common interests or goals. This can lead to long-lasting relationships and connections.

4. **Personal Growth:**

The app can provide opportunities for personal growth and development by exposing users to new ideas, perspectives, and resources.

5. **Learning Resources:**

The app can provide a central location for learning resources, making it easier for users to find relevant and helpful information.

6. **User Engagement:**

By allowing users to follow and connect with others who provide valuable content, the app can increase user engagement and retention.

7. **Brand Building:**

If the app becomes popular, it can become a valuable branding opportunity for businesses or organizations that are involved in education or professional development.

Overall, a comprehensive literature survey for a student app should cover a range of topics related to education, technology, and user behavior. The goal is to gather insights and information that can inform the design and development of an app that is effective, user-friendly, and engaging for students.

CHAPTER III

METHODOLOGY

3.1 Background / Overview of Methodology

The methodology for creating a mobile application that allows individuals at similar levels to connect and share information is based on the following steps:

Conducting user research:

The first step in creating this mobile application would be to conduct user research to understand the needs, preferences, and pain points of the target audience, which includes students and professionals.

Defining user personas:

Based on the research findings, user personas will be defined that represent the target audience. These personas will help in understanding the needs of the users and in developing features that meet their requirements.

Developing wireframes:

Once the user personas are defined, the next step would be to create wireframes for the mobile application. The wireframes will define the layout, structure, and flow of the application.

Creating a prototype:

A prototype will be developed based on the wireframes, which will help in testing the usability and functionality of the application.

Developing the application:

Once the prototype is tested and refined, the development of the application will begin. The application will be developed using agile methodology, which allows for continuous feedback and improvements.

Testing and quality assurance:

The application will be tested for usability, functionality, and quality assurance. This step will ensure that the application meets the requirements of the target audience and is free of bugs and errors.

Launching and marketing:

The final step is to launch the application and market it to the target audience. This will involve creating a marketing plan, which includes social media marketing, search engine optimization, and paid advertising.

Overall, the methodology for creating the mobile application is based on user-centric design principles and agile development methodology, which allows for continuous feedback and improvements based on user needs and preferences. The application will provide a platform for students and professionals to connect and share information, while also providing access to learning materials and resources that are most useful and relevant to them.

3.2 Project Platforms used in Project.

Flutter is a cross-platform framework that is widely used for building mobile applications for both Android and iOS platforms. Flutter allows developers to write code once and deploy it across different platforms, which makes it a popular choice for building mobile applications.

Some of the key benefits of using Flutter for mobile application development include:

1. **Hot Reload:** Flutter has a feature called Hot Reload, which allows developers to see the changes they make to the code in real-time without having to restart the application. This makes the development process faster and more efficient.
2. **Widgets:** Flutter uses a widget-based architecture, which makes it easy for developers to create custom UI elements and reuse them across different screens and applications.
3. **Dart language:** Flutter uses the Dart programming language, which is easy to learn and use. Dart is a statically-typed language that supports features like asynchronous programming, which makes it a powerful language for building mobile applications.
4. **Performance:** Flutter is known for its fast performance, thanks to its use of a rendering engine that runs directly on the device hardware.
5. **Open-source community:** Flutter has a large and active open-source community, which means developers have access to a wide range of packages, plugins, and resources that can help speed up the development process.

Overall, Flutter is a powerful and flexible framework that is well-suited for building mobile applications for both Android and iOS platforms. Its fast performance, Hot Reload feature,

and widget-based architecture make it an attractive choice for developers looking to build high-quality mobile applications.

Flutter is a popular platform for mobile application development because it offers many advantages over other platforms. Here are some of the key reasons why Flutter is a great choice for building mobile applications:

1. **Fast development:** Flutter allows for faster app development compared to other platforms, thanks to its hot reload feature, which allows developers to see changes made to the code in real-time without having to restart the application.
2. **Cross-platform compatibility:** Flutter is a cross-platform framework, which means that a single codebase can be used to build applications for both iOS and Android platforms, reducing development time and cost.
3. **Rich widget library:** Flutter offers a rich set of customizable widgets that allow developers to create beautiful, responsive, and user-friendly UI designs for their applications.
4. **High performance:** Flutter uses a compiled programming language called Dart, which is optimized for performance. This, coupled with its built-in support for hardware acceleration, makes Flutter apps fast and responsive.
5. **Open-source community:** Flutter has a large and active open-source community, which means developers have access to a wealth of resources, plugins, and libraries that can help speed up development.
6. **Google support:** Flutter is developed by Google, which means it is constantly updated and improved with new features and capabilities.

Overall, Flutter is a great choice for mobile application development due to its fast development, cross-platform compatibility, rich widget library, high performance, open-source community, and Google support.

A Flutter student app can be designed to offer a range of features to students that can enhance their learning experience. Here are some of the key features that a Flutter student app can include:

1. **Course materials:** The app can offer access to course materials such as lecture notes, readings, and videos, which students can access from anywhere and at any time.
2. **Quizzes and assessments:** The app can offer quizzes and assessments to students, allowing them to test their knowledge and get immediate feedback on their performance.

3. **Interactive activities:** The app can offer interactive activities such as gamification, simulations, and case studies, to make learning more engaging and fun.
4. **Discussion forums:** The app can offer discussion forums where students can interact with their peers and instructors to ask questions, discuss ideas, and share information.
5. **Personalized learning:** The app can offer personalized learning paths to students, allowing them to choose their own learning journey based on their strengths and weaknesses.
6. **Push notifications:** The app can offer push notifications to students to remind them of upcoming deadlines, assignments, and exams.
7. **Progress tracking:** The app can offer progress tracking to students, allowing them to monitor their learning progress and identify areas that need improvement.

Overall, a Flutter student app can offer a range of features that can enhance the learning experience for students, making education more accessible, engaging, and effective.

3.3 Proposed Methodology.



Fig no: 3.1 Agile methodology diagram.

Agile methodology is a project management approach that emphasizes flexibility, collaboration, and customer satisfaction. It was originally developed for software development but can be applied to various types of projects.

The key principles of Agile methodology include:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Agile methodology is characterized by short development cycles, called sprints, where teams work on specific tasks and deliverables. The progress is monitored through daily stand-up meetings and frequent demonstrations of working software.

The Agile approach encourages teams to adapt and respond to changes in project requirements, customer feedback, and market conditions. It also emphasizes the importance of continuous improvement and learning.

Here is a proposed methodology for developing a student base application using Flutter:

1. **Define requirements:** The first step is to define the requirements of the application. This includes identifying the target audience, defining the scope of the application, and identifying the features and functionalities that the application will offer.
2. **Design UI/UX:** The next step is to design the user interface and user experience of the application. This includes creating wireframes, designing the user interface, and defining the user flow.
3. **Develop the backend:** Once the UI/UX design is complete, the next step is to develop the backend of the application. This includes setting up a database, creating APIs, and developing server-side code.
4. **Develop the frontend:** After the backend is complete, the next step is to develop the frontend of the application using Flutter. This includes implementing the UI design, integrating APIs, and developing client-side code.
5. **Testing:** Once the frontend and backend are complete, the next step is to test the application thoroughly to ensure that it is bug-free, user-friendly, and meets all the requirements.
6. **Deployment:** After testing, the application can be deployed to the respective app stores for users to download and use.
7. **Maintenance:** Once the application is deployed, it is important to maintain it by fixing bugs, adding new features, and updating it regularly to ensure that it remains relevant and up-to-date.

Overall, this methodology involves a systematic and structured approach to developing a student base application using Flutter, ensuring that it is user-friendly, reliable, and meets all the requirements of the target audience.

3.4 Project Modules.

- Login / sign-up module.
- Home module.
- Q/A module.
- Upload Screen.
- Material module.
- Profile module.

Here are some sample modules for an app that works on Material design and includes question and answer and post features:

❖ **Authentication Module**

Includes user registration and login using email and password or social media authentication.

here is an authentication module for a Flutter app that uses Firebase:

- **Firebase Authentication Setup** - First, create a Firebase project and set up Firebase Authentication. Enable email and password authentication and any additional authentication methods you would like to offer, such as Google, Facebook, or Apple.
- **Sign In Screen** - Create a sign-in screen where users can enter their email and password or sign in with their preferred authentication provider. Use the Firebase Authentication API to handle sign-in and sign-out requests.
- **Sign Up Screen** - Create a sign-up screen where users can create a new account by entering their email and password. Use the Firebase Authentication API to create a new user account.
- **Reset Password Screen** - Create a reset password screen where users can reset their password by entering their email address. Use the Firebase Authentication API to send a password reset email to the user.
- **User Profile** - After a user has signed in, display their name and profile picture on the app's home screen. Use the Firebase Authentication API to retrieve the user's profile information.
- **Persist User Authentication State** - Use the Firebase Authentication API to persist the user's authentication state across app restarts and ensure that the user remains signed in until they sign out.
- **Error Handling** - Handle authentication errors, such as invalid email or password, account already exists, and network errors, and display appropriate error messages to the user.

- **Social Media Authentication** - If you have enabled social media authentication in Firebase Authentication, create buttons on the sign-in screen to allow users to sign in with their preferred social media account.

❖ **Material Design Module**

Implements Material Design guidelines for UI/UX, including app navigation, color schemes, typography, and UI elements.

Here's a sample Material Design module for a student app:

- **Navigation Drawer** - Implement a navigation drawer that allows users to navigate between different screens of the app. Include menu items such as Home, Profile, Notifications, Settings, and Logout.
- **App Bar** - Include an app bar that displays the app title and a search bar for searching for content within the app.
- **Color Scheme** - Use a color scheme that is appropriate for a student app. Consider using colors that are associated with learning, such as blue, green, and yellow.
- **Typography** - Use typography that is easy to read and appropriate for a student app. Consider using sans-serif fonts such as Roboto or Open Sans.
- **Cards** - Use cards to display content such as announcements, class schedules, and assignments. Cards should have a consistent design and should be easy to scan.
- **Buttons** - Use Material Design buttons to create a consistent look and feel across the app. Use primary and secondary colors to differentiate between different types of buttons.
- **Icons** - Use Material Design icons to enhance the user experience and make it easier for users to understand the app's functionality.
- **Animations** - Use animations to provide feedback to users and make the app feel more engaging. Consider using animations such as ripple effects, sliding transitions, and floating action buttons.
- **Lists** - Use Material Design lists to display content such as class schedules, assignments, and grades. Lists should have a consistent design and should be easy to read.
- **Bottom Navigation Bar** - Use a bottom navigation bar to allow users to switch between different screens of the app. Include icons and labels to make it easier for users to understand the app's functionality.

❖ Q&A Module

Enables users to ask and answer questions related to a particular topic, category, or keyword. This module includes features such as search, filters, and sorting options.

Here's a sample Q/A module for a student app:

- **Question Feed** - Display a feed of questions asked by other students, sorted by most recent. Allow users to filter questions by category or subject.
- **Ask a Question** - Allow users to ask a question by entering a title and description. Offer a selection of categories or subjects to help categorize the question.
- **Answer a Question** - Allow users to answer a question by entering a response. Allow other users to upvote or downvote answers to help surface the most helpful responses.
- **Comment on a Question or Answer** - Allow users to comment on questions or answers to ask for clarification or provide additional context.
- **Flag Inappropriate Content** - Allow users to flag questions or answers that contain inappropriate content or violate community guidelines.
- **Search for Questions** - Allow users to search for questions by keyword or category. Use search suggestions to help users find related questions.
- **Notifications** - Notify users when a question they asked has been answered or when someone has commented on their question or answer.
- **User Profiles** - Allow users to view other users' profiles to see their activity and contributions to the Q/A community.
- **Moderation Tools** - Provide moderators with tools to review flagged content and take action if necessary, such as removing content or suspending users.
- **Gamification** - Implement gamification features, such as badges or points, to encourage users to participate in the Q/A community and contribute helpful answers.

❖ Post Module

Allows users to create and publish posts on various topics, including text, images, and videos. This module includes features such as commenting, sharing, and rating.

Here's a sample Post module for a student app that allows users to upload regular posts and materials:

- **Create a Post** - Allow users to create a post by entering a title, description, and attaching any relevant files or materials, such as PDFs or images.
- **Post Feed** - Display a feed of posts created by other students, sorted by most recent. Allow users to filter posts by category or subject.
- **Like and Comment on Posts** - Allow users to like and comment on posts to engage with the content and provide feedback.
- **Search for Posts** - Allow users to search for posts by keyword or category. Use search suggestions to help users find related posts.
- **Notifications** - Notify users when someone has liked or commented on their post.
- **User Profiles** - Allow users to view other users' profiles to see their activity and contributions to the community.
- **Material Library** - Allow users to upload and browse a library of materials, such as lecture notes or study guides, organized by subject or category.
- **File Management** - Provide users with tools to manage their uploaded files, such as the ability to edit or delete their posts.
- **Moderation Tools** - Provide moderators with tools to review flagged content and take action if necessary, such as removing content or suspending users.
- **Gamification** - Implement gamification features, such as badges or points, to encourage users to participate in the community and contribute helpful posts and materials.

❖ Profile Module

Enables users to view and manage their profile information, including name, profile picture, and bio.

- **User Information** - Display the user's name, profile picture, email address, and any other relevant information, such as their major or year of study.
- **Personal Information** - Allow users to enter and update their personal information, such as their name, profile picture, email address, phone number, and other relevant details.
- **Education Information** - Allow users to enter and update their education information, such as their school, major, GPA, and other relevant details.
- **Work Experience** - Allow users to enter and update their work experience, such as their job title, company name, and work history.
- **Skills** - Allow users to enter and update their skills, such as their programming languages, tools, or other relevant skills.
- **Certifications and Awards** - Allow users to enter and update their certifications and awards, such as their certifications, awards, and other achievements.
- **Portfolio** - Allow users to showcase their work or projects by uploading documents, images, or other media.
- **Connections** - Allow users to connect with other users on the app, such as classmates, professors, or colleagues.
- **Activity Feed** - Display a feed of the user's recent activity on the app, such as their posts, comments, or other contributions.
- **Notifications** - Notify users of any relevant activity on their profile, such as new connections or updates to their portfolio.
- **Privacy Settings** - Allow users to control the visibility of their profile information and adjust their privacy settings accordingly.

- **Summary of Activity** - Display a summary of the user's activity on the app, including the number of questions they've asked, the number of answers they've given, and any other relevant metrics.
- **Questions Asked** - Display a list of questions asked by the user, sorted by most recent. Allow other users to upvote or downvote the questions to help surface the most helpful content.
- **Answers Given** - Display a list of answers given by the user, sorted by most recent. Allow other users to upvote or downvote the answers to help surface the most helpful content.
- **Badges and Achievements** - Display any badges or achievements earned by the user for their contributions to the Q/A module or other modules of the app.
- **Edit Profile** - Allow users to edit their profile information, such as their profile picture or email address.
- **Settings** - Allow users to adjust their app settings, such as notification preferences or language settings.
- **Contact Support** - Provide users with a way to contact support if they have any issues or questions.
- **Logout** - Allow users to log out of the app.

3.5 Diagrams:

Here are some diagrams that could be useful for a student app that contains a material module, Q/A module, and post module:

3.5.1 Use Case Diagram:

This diagram shows the actors and the use cases associated with the student app. Actors could include students, professors, or moderators, and use cases could include creating posts, searching for materials, or answering questions.

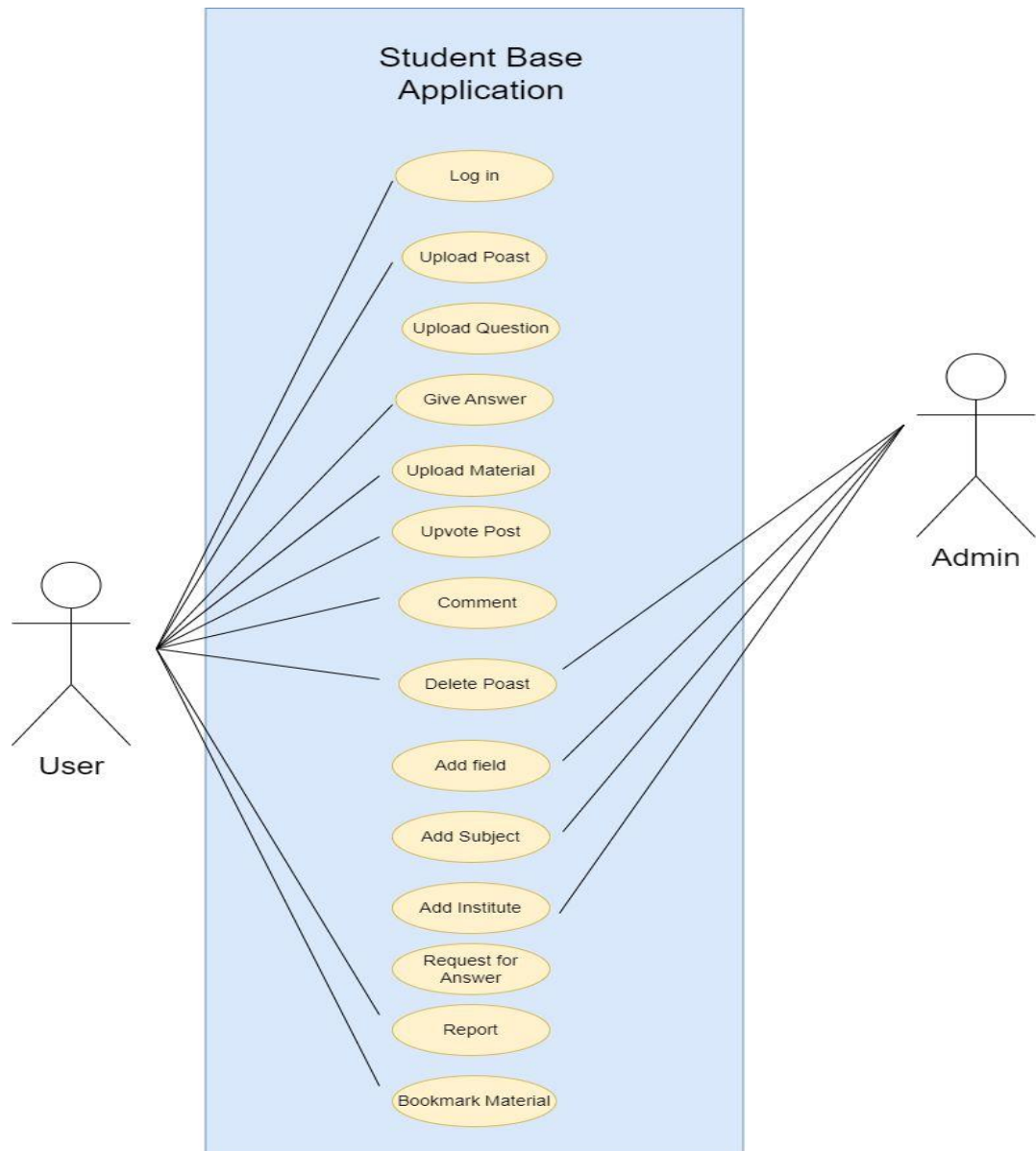


Fig no: 3.2 Use case diagram.

3.5.2 ER Diagram:

The ER diagram for a Student Hub that works on material, questions and answers, and posts is a graphical representation of the database schema that defines the entities and relationships between them. The ER diagram consists of boxes representing entities, lines representing relationships, and arrows indicating the direction of the relationships.

In this ER diagram, we have five entities: Material, Subject, Student, Question, Answer, and Post. Material represents the study material that is uploaded by teachers or professors for students to access. Subject represents the different subjects that are taught in the school or college. Student represents the students who use the application to access study material, ask questions, post answers, and create posts. Question represents the questions asked by students related to a particular subject, and Answer represents the answers given by students related to a particular question. Post represents the posts created by students on a particular topic.

The entities are connected through relationships, which are represented by lines. The relationships between entities are determined by the cardinality and optionality of the relationships. For example, a Material entity can be associated with multiple Subject entities, but each Subject entity can be associated with only one Material entity. Similarly, a Student entity can ask multiple Question entities, but each Question entity can be associated with only one Student entity. An Answer entity can be associated with only one Question entity, but each Question entity can have multiple Answer entities associated with it.

The direction of the relationships is indicated by the arrows. For example, the arrow pointing from the Question entity to the Student entity indicates that a Question entity is associated with a Student entity. The arrow pointing from the Student entity to the Question entity indicates that a Student entity can ask multiple Question entities.

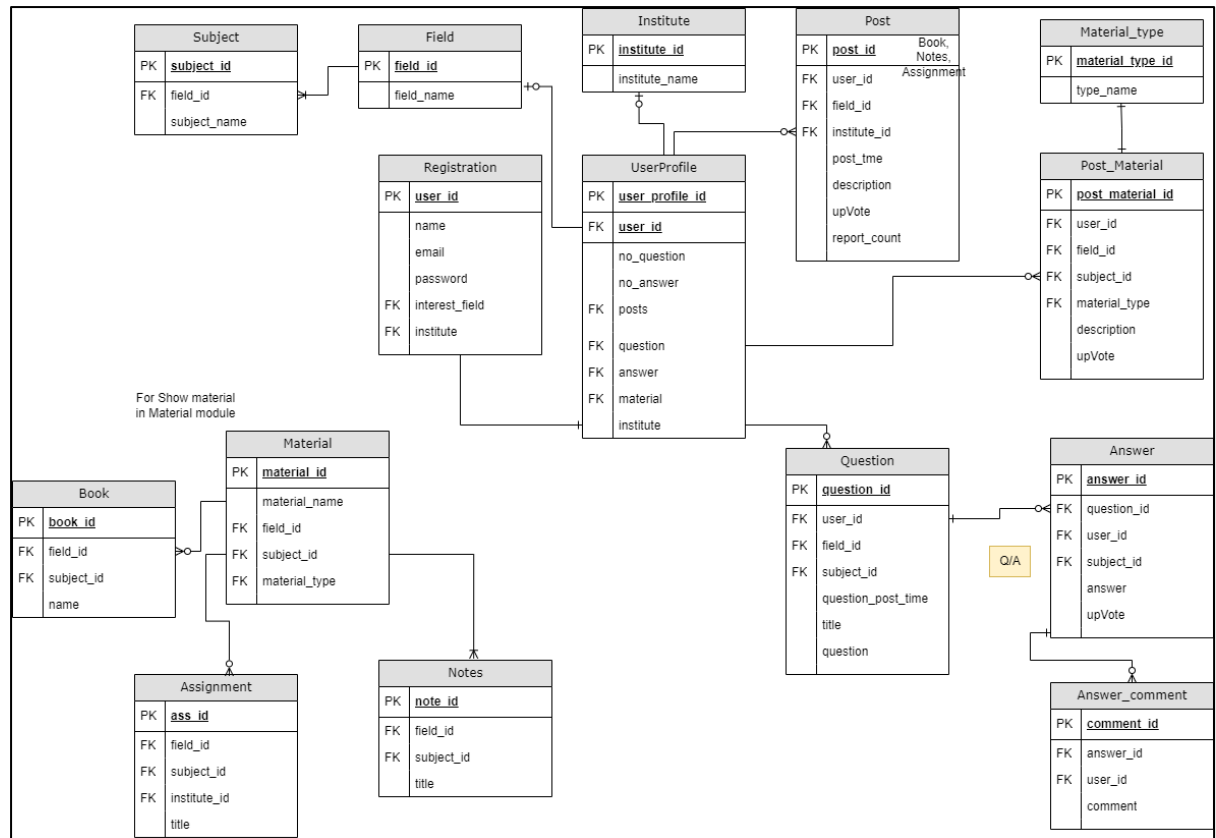


Fig no: 3.3 ER Diagram

CHAPTER IV

Implementation

4.1 Main Functions with explanation

4.1.1 Functionality for Material:

here are some functions that can be used in a Student Hub to work with materials:

- **load_material()**: This function loads the material from a file or database and returns it as a list of dictionaries, where each dictionary represents a piece of material and contains information such as the title, description, and content of the material.
- **display_material(material)**: This function takes in the list of material and displays it to the user in a readable format, such as printing the title and description of each piece of material.
- **add_material(material)**: This function allows the user to add a new piece of material to the list. It prompts the user to enter the title, description, and content of the material, creates a new dictionary with this information, and appends it to the list.
- **edit_material(material)**: This function allows the user to edit an existing piece of material. It prompts the user to select the material they want to edit, and then prompts them to enter the new title, description, and content. It then updates the corresponding dictionary in the list.
- **delete_material(material)**: This function allows the user to delete an existing piece of material. It prompts the user to select the material they want to delete, and then removes the corresponding dictionary from the list.
- **search_material(material, query)**: This function allows the user to search for material that contains a certain keyword or phrase. It takes in the list of material and a search query, and returns a list of dictionaries that match the query.
- **save_material(material)**: This function saves the list of material to a file or database so that it can be loaded again later.

These functions can be combined to create a comprehensive material management system for a Student Hub

4.1.2 Functions for Question and Answers:

Here are some main functions for a Student Hub that works on questions and answers:

Ask Questions:

Allow students to ask questions related to their studies. This function should allow students to provide details of the question, such as the subject or topic, level of difficulty, and any supporting materials.

Answer Questions: Allow students to answer questions asked by other students. This function should enable students to provide detailed and informative answers that are relevant to the question.

Upvote/Downvote Questions and Answers: Allow students to upvote or downvote questions and answers posted by other students. This function will help to ensure that the most relevant and informative questions and answers are easily accessible to other students.

Search Questions: Enable students to search for questions that have been previously asked and answered on the platform. This function should allow students to filter questions by subject, topic, and difficulty level.

Notifications: Notify students when their questions have been answered or when new questions related to their studies have been posted. This function should ensure that students are always up-to-date with the latest information related to their studies.

Commenting: Allow students to comment on questions and answers posted by other students. This function will help to facilitate discussions and enable students to provide additional context or clarification where necessary.

Expert Support: Provide expert support to students where necessary. This function should allow students to access additional resources and support from qualified professionals in their field of study.

4.1.3 Functionality for Posts:

Here are some main features and functions for a Student Hub that works on posts:

Create Posts: Allow students to create posts related to their studies, such as notes, study tips, or resource recommendations. This function should allow students to add images, videos, or other multimedia content to their posts.

Like and Comment: Allow students to like and comment on posts created by other students. This function will help to encourage engagement and foster a sense of community within the platform.

Search Posts: Enable students to search for posts related to their studies. This function should allow students to filter posts by subject, topic, and level of difficulty.

Notifications: Notify students when new posts related to their studies have been added. This function should ensure that students are always up-to-date with the latest information related to their studies.

Save Posts: Allow students to save posts that they find particularly helpful or informative. This function should enable students to easily access these resources at a later time.

Share Posts: Enable students to share posts with their peers on social media or other platforms. This function will help to expand the reach of useful resources and facilitate knowledge sharing among students.

Moderation: Implement moderation features to ensure that all posts are appropriate and relevant to the platform. This function should include the ability to flag inappropriate posts or comments for review by an administrator.

Analytics: Provide analytics and insights to help students and administrators understand how posts are being used and engaged with on the platform. This function should include metrics such as views, likes, comments, and shares.

4.2 Non-functional requirements:

1. **Performance:** The application should be able to handle a large number of users and data without significant lag or crashes.
2. **Scalability:** The application should be able to scale up or down depending on the number of users and data.
3. **Security:** The application should be designed with security in mind, including authentication and authorization mechanisms to protect user data and prevent unauthorized access.
4. **Availability:** The application should be available 24/7 and have a reliable backup and recovery mechanism in case of any failures or disruptions.
5. **User Experience:** The application should be intuitive and easy to use, with a clean and attractive design, and should provide a smooth and seamless user experience.
6. **Accessibility:** The application should be accessible to users with disabilities, with features such as screen reader compatibility and keyboard navigation.
7. **Compatibility:** The application should be compatible with different mobile devices and operating systems, such as Android and iOS.
8. **Data Privacy:** The application should comply with data privacy laws and regulations, such as GDPR and CCPA, and should provide users with clear and transparent information about how their data is collected, used, and shared.
9. **Maintainability:** The application should be easy to maintain and update, with a modular and well-documented codebase, and should be designed with future changes and enhancements in mind.
10. **Performance metrics:** The application should have monitoring and reporting mechanisms in place to track and analyse its performance metrics, such as response time, resource usage, and user engagement.

4.3 Code with Explanation:

1) Authentication Methods:

getUserDetails(): This method is use to take current user details.

signUpUser(): This method is call when new user want to create new account

loginUser(): This method is call when user login in app.

signOut(): This method is used for signing out of user.

clearCache(): When user sign out from application then this method is call in order to remove application cache.

Code:

```
class AuthMethods {
    final FirebaseAuth _auth = FirebaseAuth.instance;
    final FirebaseFirestore _firestore = FirebaseFirestore.instance;

    // get user details
    Future<model.User> getUserDetails() async {
        User currentUser = _auth.currentUser!;

        // Take current user toke id.
        DocumentSnapshot snap = await
        _firestore.collection('users').doc(currentUser.uid).get();
        // -> Take snap of current user & return it.
        return model.User.fromSnap(snap);
    }

    // signup user
    Future<String> signUpUser({ // data type is future because all the call of firebase is
    asynchronous
        required String email,
        required String password,
        required String username,
        required String institue,
        required String field,
        required Uint8List file
    }) async{
        String res = "Some error occured";
        try {
            if(email.isNotEmpty || password.isNotEmpty || username.isNotEmpty ||
            institue.isNotEmpty || field.isNotEmpty) {
                // register user
                UserCredential cred = await _auth.createUserWithEmailAndPassword(email: email,
                password: password);
```


print(cred.user!.uid); // give user id // user! ---> user can be return as null. That's way put ! mark.

String photoUrl = await StorageMethods().uploadImageToStorage('profilePics', file, false);

```
// add user to our firestore database
// --> 2nd method using Convert into JSON
model.User user = model.User(
  username : username,
  uid : cred.user!.uid,
  email : email,
  institue: institue,
  field: field,
  followers : [],
  following : [],
  photoUrl : photoUrl,
);
await _firestore.collection('users').doc(cred.user!.uid).set(user.toJson(),);
res = "success";
}
} catch(err) {
  res = err.toString();
}
return res;
}

// logging in user
Future<String> loginUser({
  required String email,
  required String password
}) async{
  String res = "Some error occured";

  try {
    if(email.isNotEmpty || password.isNotEmpty) {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      res = "success";
    } else {
      res = "Please enter all the fields";
    }
  } catch(err) {
    res = err.toString();
  }
  return res;
}
```

```
// Sign out.
Future<void> signOut() async {
  await _auth.signOut();
  clearCache();
}

// --> Remove cahched when user log out
// Function to clear the cache
Future<void> clearCache() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  await prefs.clear();
}
}
```

2) Firestore Methods:

uploadPost(): This method is used to upload any post in firebase storage.

uploadPdf(): This method used when student upload any material which extension is .pdf file.

uploadQuestion(): This method call when user upload any question in Q/A module.

uploadAnswer(): This method is call when user answer the selected question.

likePost(): This method is call when any user like or dislike the post, material or answer.

postComment(): This method is call wen user post any comment in post feed or in answer feed.

Code:

```
class FirestoreMethods {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  // Upload post
  Future<String> uploadPost(
    String description,
    Uint8List file,
    String uid,
    String username,
    String profImage,
    String userField,
    String userInstitue
  ) async {
    String res = "Some error occured";
    try {
```

```
// upload in firestore & take url of image & store in variable.
String photUrl = await StorageMethods().uploadImageToStorage('posts', file, true);

String postId = const Uuid().v1();
Post post = Post(
  description: description,
  uid: uid,
  username: username,
  postId: postId,
  datePublished: DateTime.now(),
  postUrl: photUrl,
  profImage: profImage,
  likes: [],
  field: userField,
  institute: userInstitute);
_firestore.collection('posts').doc(postId).set(post.toJson(),);
res = "success";
} catch(err) {
  res = err.toString();
}
return res;
}
// --> Upload pdf
Future<String> uploadPdf(
  String fieldId,
  String subjecId,
  String title,
  String materialType,
  String materialField,
  String materialSubject,
  String materialUrl,
  // Uint8List file,
  String uid,
  String username,
  String profImage,
  String userField,
  String userInstitute
) async {
String res = "Some error occurred";
try {
  String materialId = const Uuid().v1();
  MaterialTwo material = MaterialTwo(
    title: title,
    materialId: materialId,
    materialType: materialType,
    materialSubject: materialSubject,
```

```

        materialField: materialField,
        materialUrl: materialUrl,
        uid: uid,
        username: username,
        profImage: profImage,
        userField: userField,
        userInstitute: userInstitute,
        datePublished: DateTime.now(),
        likes: [],
    );
    _firestore.collection('fields').doc(fieldId).collection('Subjects').doc(subjectId).collection(materialType).doc(materialId).set(material.toJson(),);
    res = "success";
  } catch(err) {
    res = err.toString();
  }

  return res;
}

// --> Upload question
Future<String> uploadQuestion(
  String question,
  String questionType,
  // File file,
  // Uint8List file,
  String photoUrl,
  String uid,
  String username,
  String profImage,
  String userField,
  String userInstitute
) async {
  String res = "Some error occurred";
  try {
    String questionId = const Uuid().v1();
    Question que = Question(
      question: question,
      questionType: questionType,
      uid: uid,
      username: username,
      questionId: questionId,
      datePublished: DateTime.now(),
      postUrl: photoUrl,
      profImage: profImage,
      // likes: [],

```

```

        field: userField,
        institute: userInstitute
    );
    _firestore.collection('questions').doc(questionId).set(que.toJson(),);
    res = "success";
} catch(err) {
    res = err.toString();
    print(err.toString());
}

return res;
}

// --> Upload Answer
Future<String> uploadAnswer(
    String questionId,
    String answer,
    // File file,
    // Uint8List file,
    String photoUrl,
    String uid,
    String username,
    String profImage,
    String userField,
    String userInstitute
) async {
    String res = "Some error occurred";
    try {
        String answerId = const Uuid().v1();
        Answer ans = Answer(
            questionId: questionId,
            answer: answer,
            uid: uid,
            username: username,
            answerId: answerId,
            datePublished: DateTime.now(),
            postUrl: photoUrl,
            profImage: profImage,
            likes: [],
            field: userField,
            institute: userInstitute
        );
        _firestore.collection('questions').doc(questionId).collection('answers').doc(answerId).set(
            ans.toJson(),);

        res = "success";
    }
}

```

```

    } catch(err) {
        res = err.toString();
        print(err.toString());
    }

    return res;
}

// --> Update likes
Future<void> likePost(String postId, String uid, List likes) async{
    try {
        if(likes.contains(uid)) { // for dislike. // then remove like (uid) from likes field in
firebase database.
            await _firestore.collection('posts').doc(postId).update({
                'likes' : FieldValue.arrayRemove([uid]),
            });
        }
        else { // for like. // then store uid in likes field (list) in firebase database.
            await _firestore.collection('posts').doc(postId).update({
                'likes' : FieldValue.arrayUnion([uid]),
            });
        }
    } catch(e) {
        print(e.toString());
    }
}

// store comment in firebase database.
Future<void> postComment(String postId, String text, String uid, String name, String
profilePic) async{
    try {
        if(text.isNotEmpty) {
            String commentId = const Uuid().v1(); // -> generate unique comment id.
            await
            _firestore.collection('posts').doc(postId).collection('comments').doc(commentId).set({
                'profilePic' : profilePic,
                'name' : name,
                'uid' : uid,
                'text' : text,
                'commentId' : commentId,
                'datePublished' : DateTime.now(),
            });
        } else {
            print("Text is empty");
        }
    } catch(e) {

```

```

        print(e.toString());
    }
}
}

```

3) Storage Methods.

uploadImageToStorage(): This method use to store images in firebase storage database and generate the url link for accessing the image.

uploadMaterialToStorage(): This method use to store materials pdf files in firebase storage and generate the url link for accessing materials files.

Code:

```

class StorageMethods {
    final FirebaseStorage _storage = FirebaseStorage.instance;
    final FirebaseAuth _auth = FirebaseAuth.instance; // for get user id.

    // adding image to firebase storage
    Future<String> uploadImageToStorage(String childName, Uint8List file, bool isPost)
    async{ // this function can use for store profile and post
        Reference ref = _storage.ref().child(childName).child(_auth.currentUser!.uid); // ref() -
        >this method point to the our file in storage. child() -> represent folder. it can be exist or
        does not exist.

        if(isPost) {
            String id = const Uuid().v1(); // generate unique id for post. // -> It can't same with
            user id.
            ref = ref.child(id); // name of post be a unique id. // --> create user unique id folder &
            store user image in that.
        }

        UploadTask uploadTask = ref.putData(file);

        TaskSnapshot snap = await uploadTask;
        String downloadUrl = await snap.ref.getDownloadURL(); // it's fetch download URL
        to the file being uploaded here

        return downloadUrl;
    }

    // adding pdf to firebase storage
    Future<String> uploadMaterialToStorage(String material, String type, File file, bool
    isPost) async{ // this function can use for store profile and post
        Reference ref = _storage.ref().child(material).child(type).child(_auth.currentUser!.uid);

```

```

    if(isPost) {
        String id = const Uuid().v1(); // generate unique id for post. // -> It can't same with
user id.
        ref = ref.child(id); // name of post be a unique id. // --> create user unique id folder &
store user image in that.
    }

    // upload pdf file to firebase storage
    TaskSnapshot taskSnapshot = await ref.putFile(file);
    String downloadUrl = await taskSnapshot.ref.getDownloadURL(); // it's fetch
download URL to the file being uploaded here

    // return the url of pdf file uploaded location.
    return downloadUrl;
}
}

```

4) User Provider:

In this code we update the user value, data when user login, log out, sign in, sign out. Here we apply the listener which help to determine the user login or not. If it's login then it provide all the user details.

Code:

```

class UserProvider with ChangeNotifier {
    User? _user; // making private field. // nulllabel.
    final AuthMethods _authMethods = AuthMethods();

    User get getUser => _user!;

    Future<void> refreshUser() async { // refresh user every time.
        User user = await _authMethods.getUserDetails();
        _user = user;
        notifyListeners(); // it notify all the listener provider k _user (gobal) data is changed so
you need to update your value.
    }
}

```


5) Persistent User.

In this code we determine that if user login then user see home screen of the app. Or if user is not login in the app then user see login screen. This will done using above method **UserProvider** which notify the all listener.

Code:

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

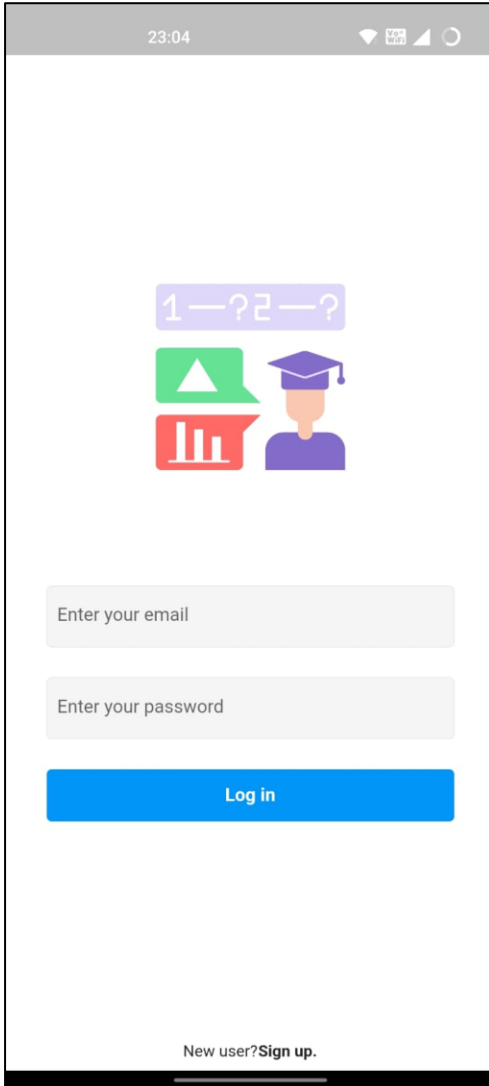
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => UserProvider(),),
      ],
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Student app',
        theme: ThemeData.light().copyWith(
          scaffoldBackgroundColor: primaryColor,
        ),
        home: StreamBuilder(
          stream: FirebaseAuth.instance.authStateChanges(), // when state change sign in in
app or sign out in app
          builder: (context, snapshot) {
            if(snapshot.connectionState == ConnectionState.active) {
              if(snapshot.hasData) { // user is authenticated.
                return const ResponsiveLayout(
                  mobileScreenLayout: MobileScreenLayout(),
                  webScreenLayout: WebScreenLayout(),
                );
              } else if (snapshot.hasError) {
                return Center(
                  child: Text('${snapshot.error}'), // some internal error occurred
                );
              }
            }
            if(snapshot.connectionState == ConnectionState.waiting) {
              return Center(
                child: CircularProgressIndicator(
                  color: primaryColor,
                ),
              );
            }
          }
        )
      )
    );
  }
}
```

```
        return const LoginScreen(); // snapshot has no data regarding user is logged in.  
    },  
    ),  
    ),  
);  
}  
}
```

CHAPTER V

RESULTS

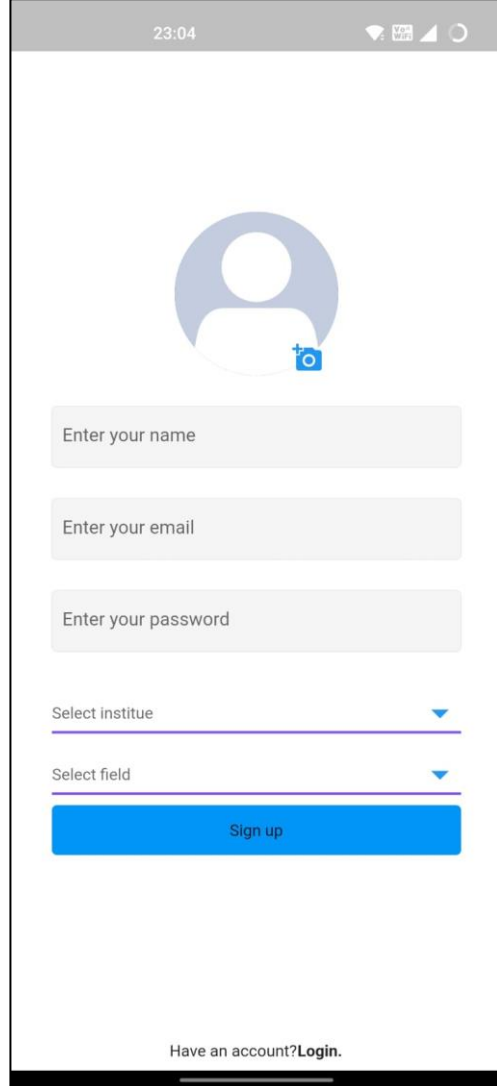
1) Login screen



The login screen features a status bar at the top with the time 23:04 and standard Android icons. The main content area includes a decorative graphic with a password field (1—?2—?), a green triangle icon, a red bar chart icon, and a purple graduation cap icon. Below this are two input fields: "Enter your email" and "Enter your password". A prominent blue "Log in" button is centered below the fields. At the bottom, a link reads "New user? Sign up."

Fig no: 5.1 Login Screen

2) sign-up screen.



The sign-up screen features a status bar at the top with the time 23:04 and standard Android icons. The main content area includes a decorative graphic with a circular profile icon and a blue camera icon. Below this are three input fields: "Enter your name", "Enter your email", and "Enter your password". This is followed by two dropdown menus labeled "Select institute" and "Select field". A prominent blue "Sign up" button is centered below the dropdowns. At the bottom, a link reads "Have an account? Login."

Fig no: 5.2 Sign up Screen

3. Feed Screen.

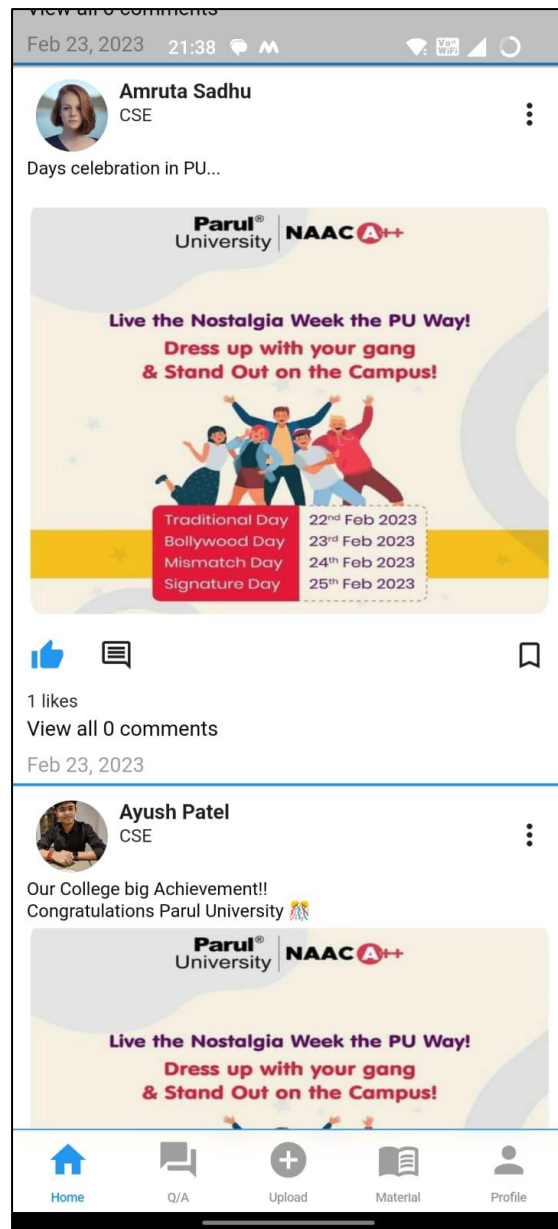


Fig no: 5.3 Feed Screen

4. Q/A Screen.

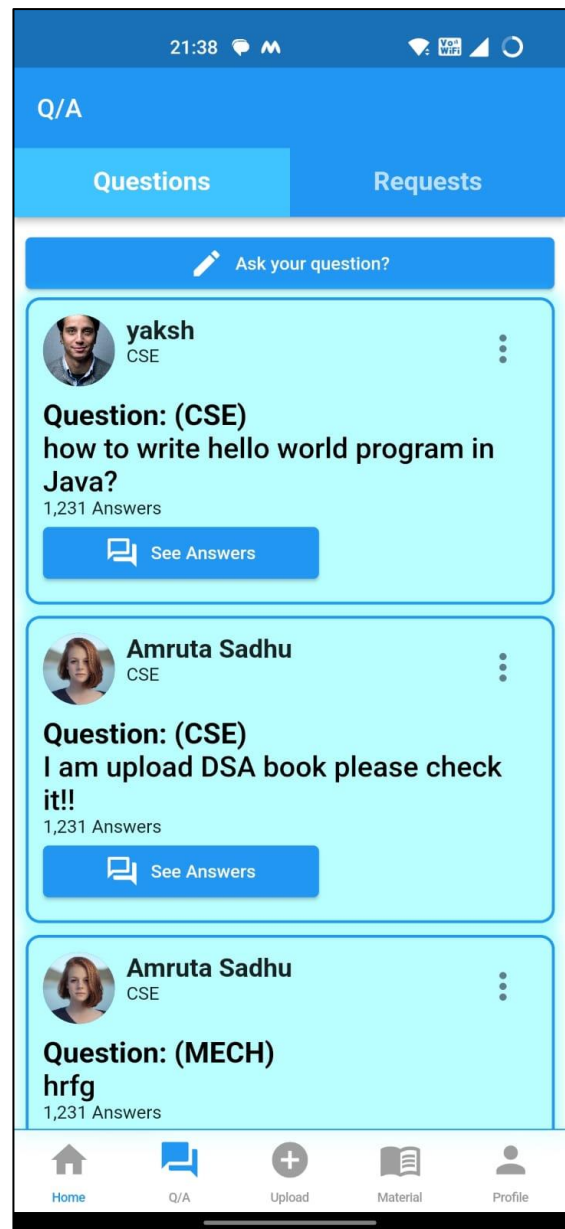


Fig no: 5.4 Questions Feed Screen

5. Answer Screen

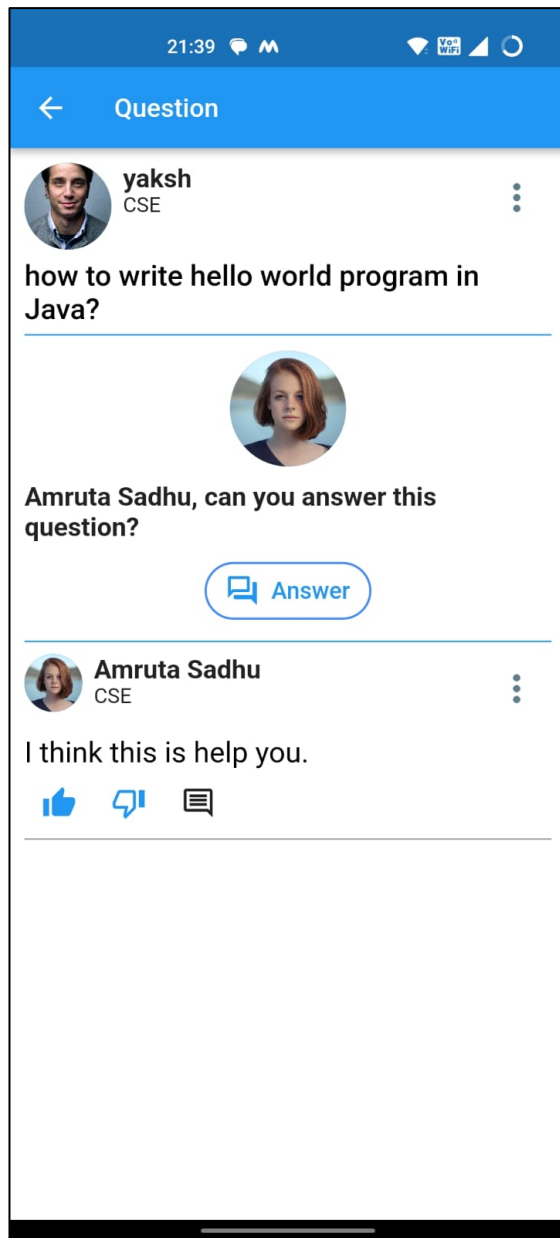


Fig no: 5.5 Answer lists Screen

6. Upload Answer Screen

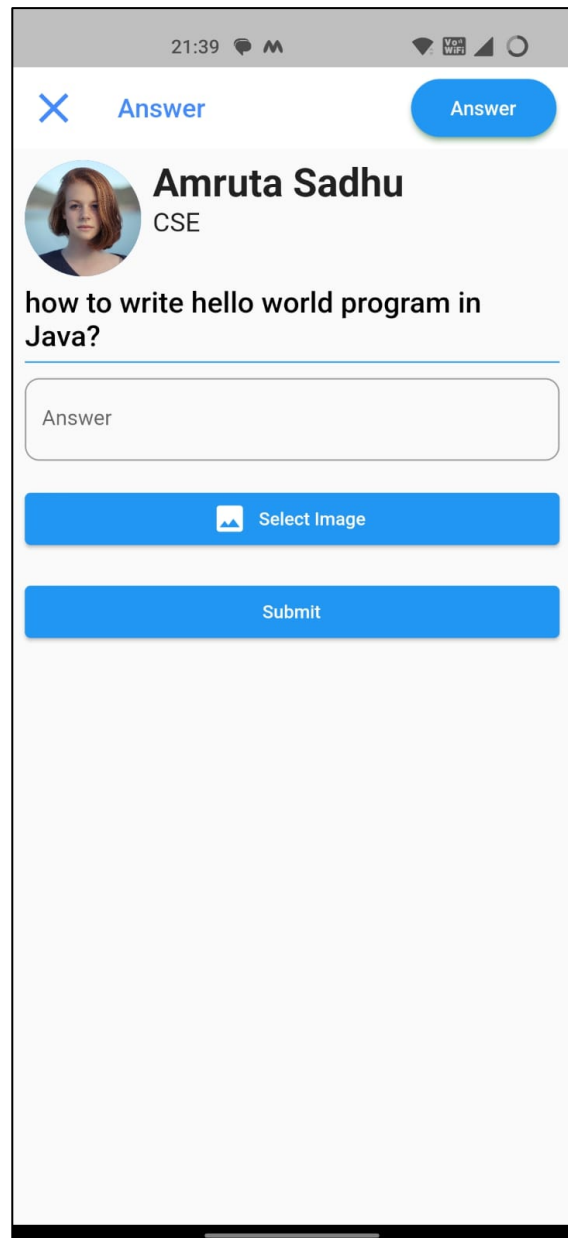


Fig no: 5.6 Upload an answer

7. See Answer in detail

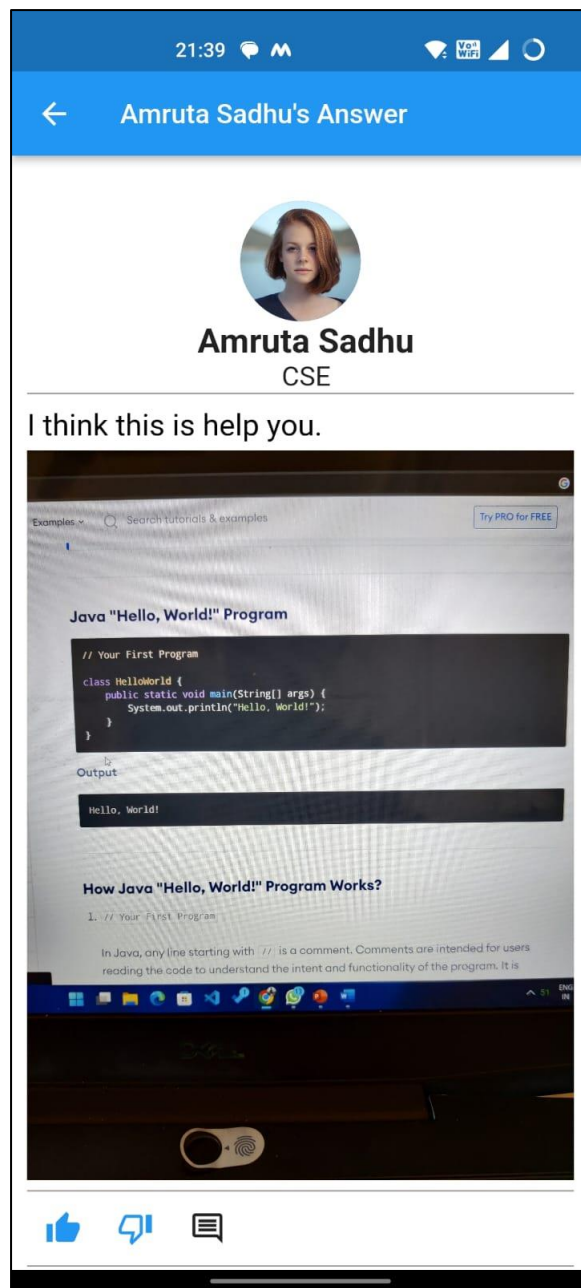


Fig no: 5.7 Detail answer screen

8. Upload Screen.

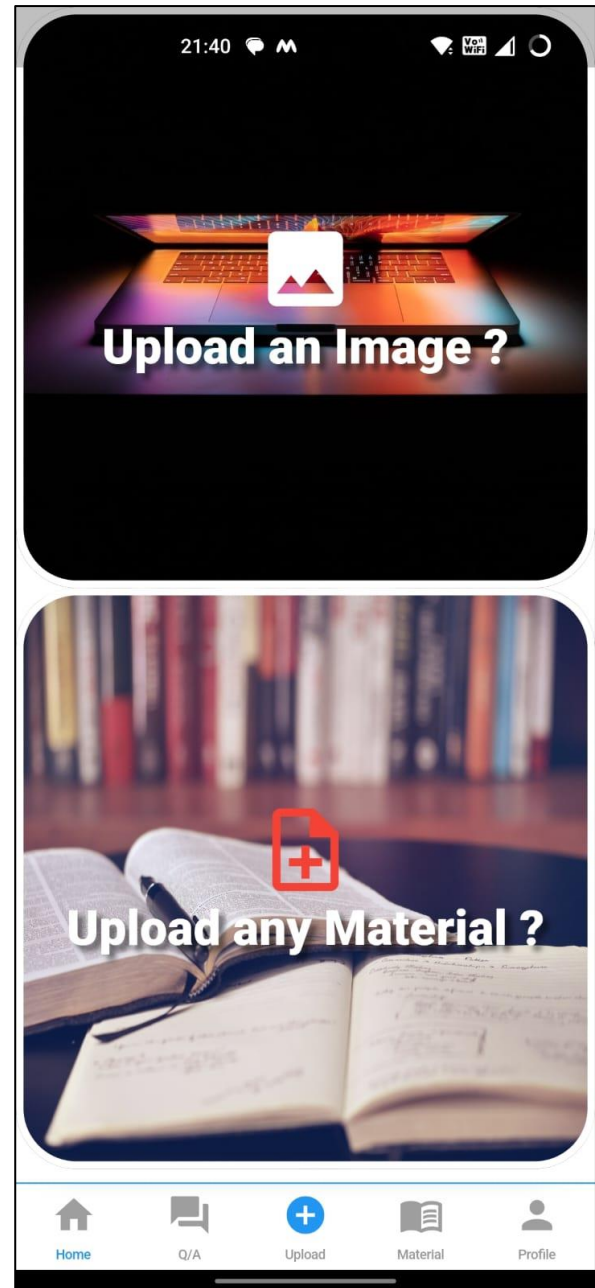


Fig no: 5.8 Upload screen

9. Upload Image (post)

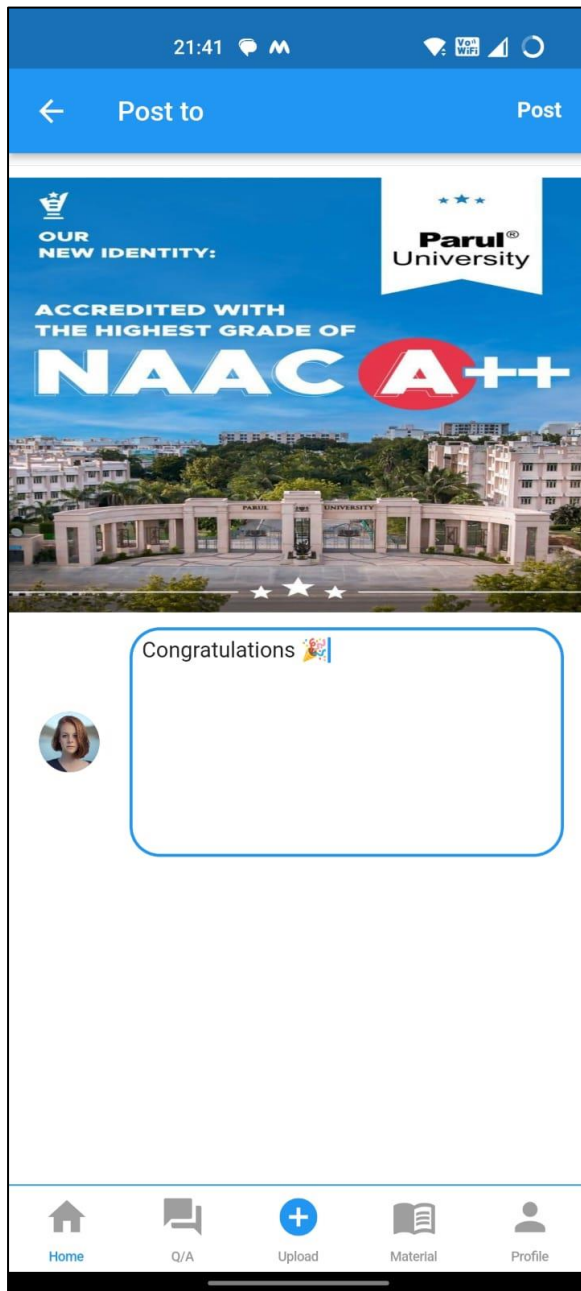


Fig no: 5.9 Post Upload

10. Upload Material

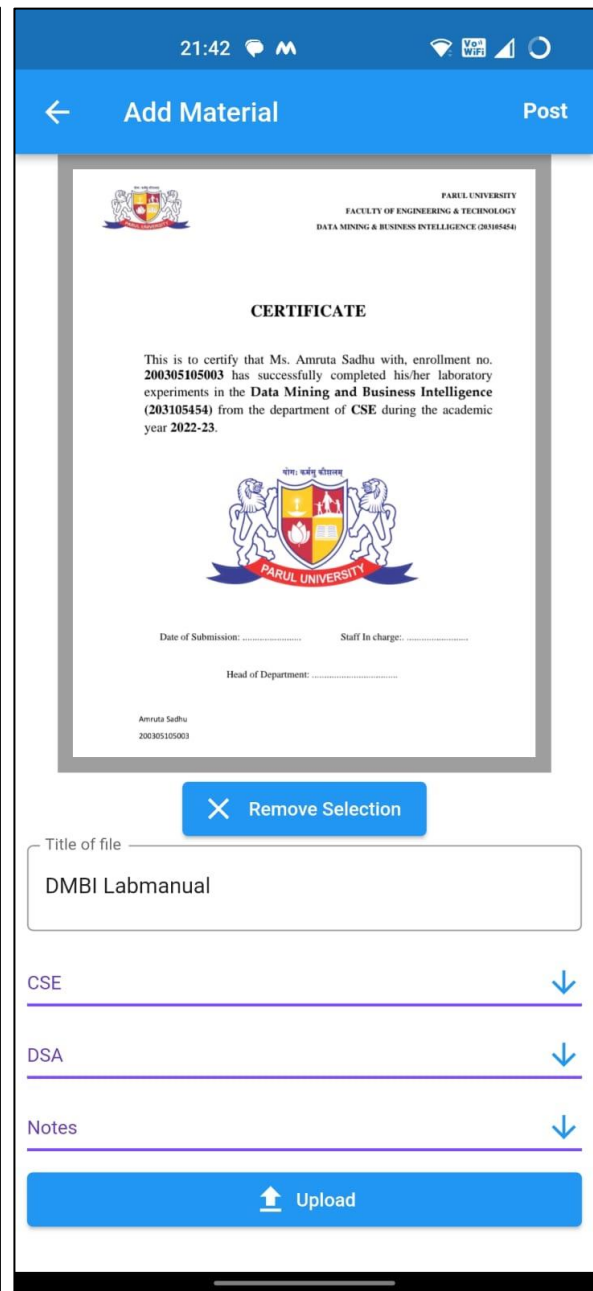


Fig no: 5.10 Material Upload

11. Upload Question

21:40

← Upload your question here.

Question Type
CSE

Question
Here I write my question.

Select Image

(Pair of 3)

Item set	frequency
I_1, I_2, I_3	3
I_1, I_2, I_4	2
I_2, I_3, I_4	2

(Pair of 4)

Item set	frequency
I_1, I_2, I_3, I_4	1

Calculate Confidence.

Permutation	Confidence
$I_1 \wedge I_2 \rightarrow I_3$	$3/4$ 75%
$I_1 \wedge I_3 \rightarrow I_2$	$3/3$ 100%
$I_2 \wedge I_3 \rightarrow I_1$	$3/4$ 75%
$I_3 \rightarrow I_1 \wedge I_2$	$3/4$ 75%
$I_2 \rightarrow I_1 \wedge I_3$	$3/5$ 60%
$I_1 \rightarrow I_2 \wedge I_3$	$3/4$ 75%
$I_1 \wedge I_2 \rightarrow I_4$	$2/4$ 50%
$I_1 \wedge I_4 \rightarrow I_2$	$2/2$ 100%

Remove Image

Submit

Fig no: 5.11 Question Upload

12. Material Screen (Field)

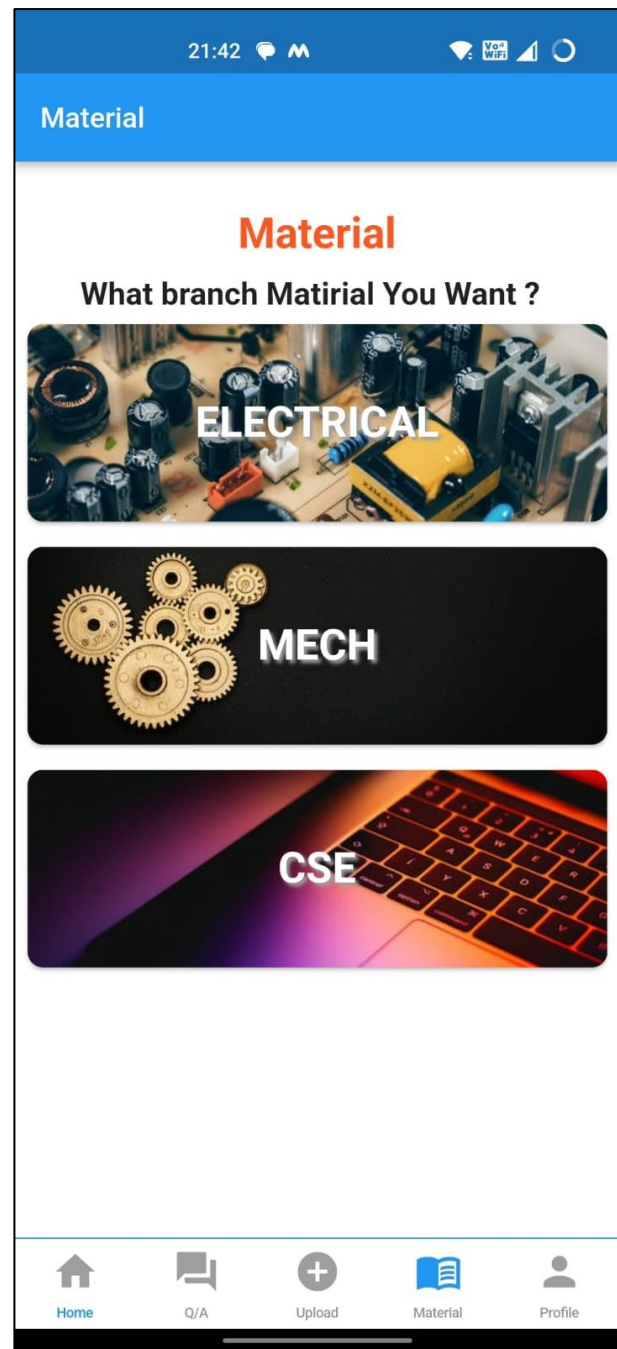


Fig no: 5.12 Field vise material screen

13. Subject Screen

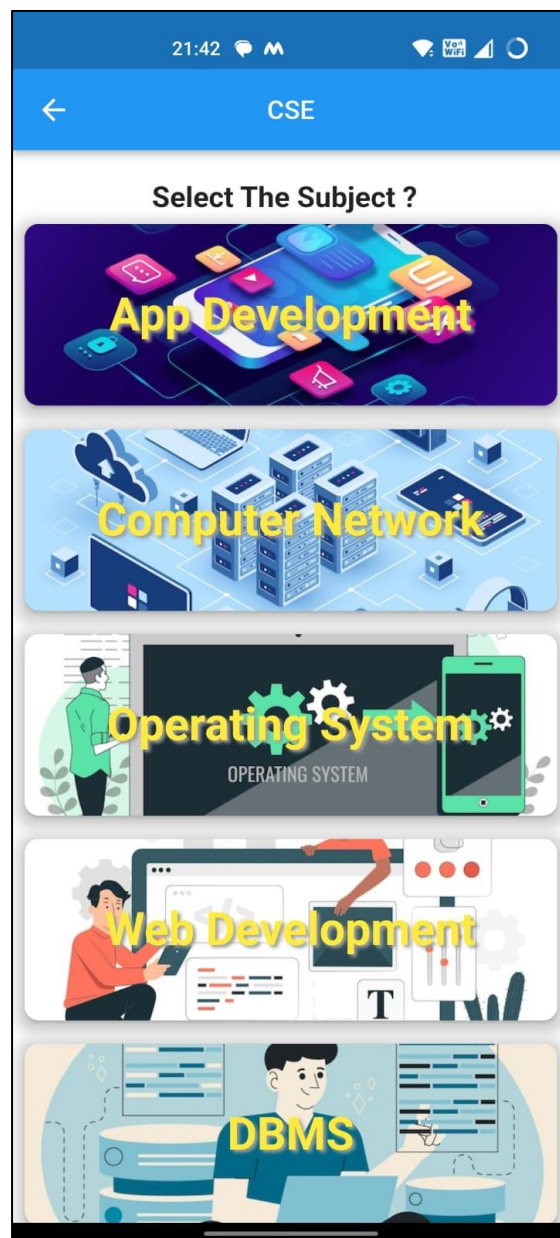


Fig no: 5.13 Subject wise material

14. Profile Screen

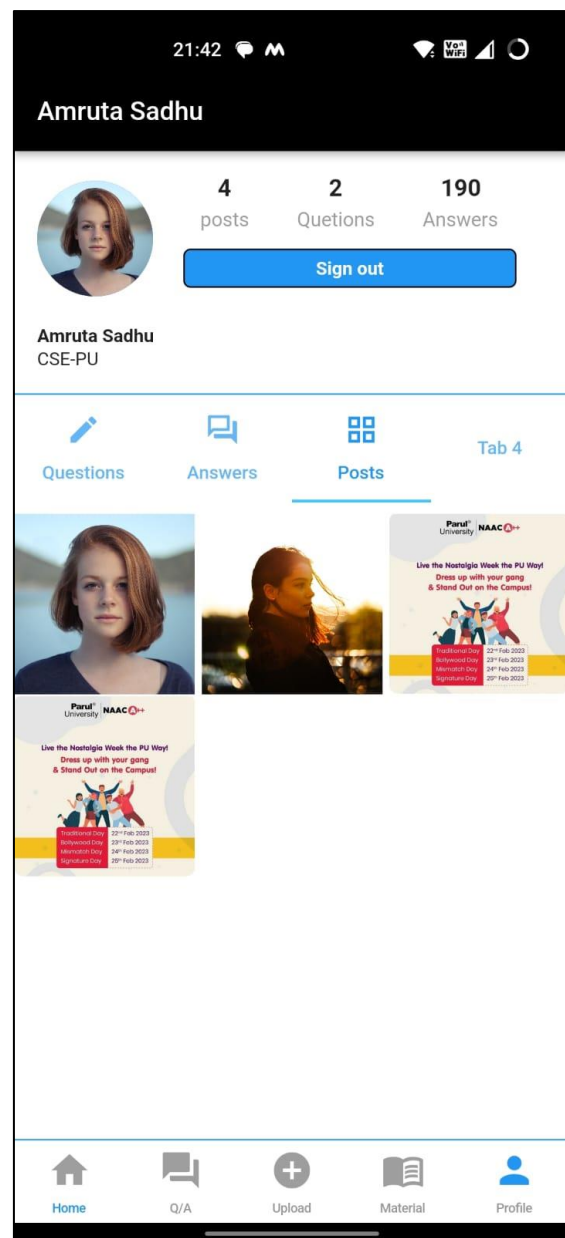


Fig no: 5.14 Profile Screen.

CHAPTER VI

USER MANUAL

6.1 SOFTWARE REQUIREMENTS:

To build a student-based application that works on material, questions, answers, and posts using Flutter and Firebase, you would need the following software requirements:

Operating System: Operating system is required. Without operating system nothing is happen. Minimum OS for Microsoft Windows is **Windows 10 or later (64-bit), x86-64 based, macOS (64-bit).**

Flutter: Flutter is a free and open-source mobile application development framework created by Google. You will need to install Flutter on your machine to develop the mobile application.

Dart Programming Language: Flutter uses the Dart programming language, which is also developed by Google. You will need to be familiar with Dart to develop the application.

Firebase: Firebase is a mobile and web application development platform developed by Google. It provides a variety of tools and services that can be used to develop mobile applications. You will need to create a Firebase project and configure it to use the required services.

Firebase Authentication: Firebase Authentication provides a simple way to authenticate users to your application. You can use Firebase Authentication to allow users to sign in to your application.

Firebase Cloud Firestore: Firebase Cloud Firestore is a NoSQL document database that allows you to store and sync data between your application clients and Firebase. You can use Cloud Firestore to store the data related to material, questions, answers, and posts.

Firebase Storage: Firebase Storage is a cloud storage service that allows you to store and retrieve user-generated content like images, videos, and audio files. You can use Firebase Storage to store images or files related to material, questions, answers, and posts.

Flutter Packages: Flutter provides a wide range of packages that you can use to develop mobile applications quickly. Some popular packages that you can use to develop the

application include the `firebase_core` package, `cloud_firestore` package, `firebase_storage` package, and `flutter_bloc` package.

UI/UX Design: To make your application look attractive and user-friendly, you will need to have a good understanding of UI/UX design principles. You can use tools like Sketch or Figma to create UI designs.

IDE: You will need an Integrated Development Environment (IDE) to write and test your Flutter code. Some popular IDEs for Flutter development include Android Studio, Visual Studio Code, and IntelliJ IDEA.

These are some of the software requirements you would need to build a student-based application that works on material, questions, answers, and posts using Flutter and Firebase.

6.2 Hardware Requirements:

The hardware requirements for developing a Student Hub that works on uploading and seeing material, asking questions, giving answers, and posting on Flutter and Firebase would include:

Computer: You would need a computer to develop the application. The computer should meet the minimum requirements for running the Flutter framework and the IDE you plan to use.

Mobile Device: You will need a mobile device to test your application on real devices. The device should have a recent version of iOS or Android and meet the minimum requirements for running Flutter apps.

Internet Connection: An active internet connection is essential to use Firebase services, such as authentication, cloud storage, and Firestore.

Storage Space: The storage space on your computer should be sufficient to store the required software, IDE, and application files.

RAM: Your computer should have a minimum of 4GB of RAM to run the Flutter framework and IDE smoothly.

Processor: A processor with a minimum of 2 GHz clock speed is recommended to handle the development environment and the mobile application's requirements.

Graphics Card: A dedicated graphics card is not required, but it can help speed up the development process, especially when working with complex UI designs.

Additional Hardware: Depending on your application's requirements, you may need additional hardware such as a physical device for testing sensors or other features, or a compatible emulator for simulating specific devices or operating systems.

These are some of the hardware requirements you would need to develop a Student Hub that works on uploading and seeing material, asking questions, giving answers, and posting on Flutter and Firebase.

6.3 STEPS TO RUN THE PROJECT

Flutter application and have set up your Firebase project, here are the general steps to run the project:

1. **Install the Flutter SDK:** First, you need to install the Flutter SDK on your computer. You can download the Flutter SDK from the official Flutter website, and follow the installation guide to install it.
2. **Set up an IDE:** Next, you need to set up an IDE to develop your Flutter application. You can use either Android Studio, IntelliJ IDEA, or Visual Studio Code. Follow the installation guide for your chosen IDE and install the Flutter and Dart plugins.
3. **Clone your project:** Clone your project from your version control system (e.g., GitHub, Bitbucket) into your local environment.
4. **Install project dependencies:** Navigate to the root of your project in the terminal and run the command `flutter pub get`. This will install all the dependencies specified in the `pubspec.yaml` file.
5. **Configure Firebase:** Configure your Firebase project by adding your Firebase configuration files to your Flutter project. You can follow the Firebase documentation to configure your project.
6. **Run the application:** Once everything is set up, you can run the application by using the command `flutter run` in the terminal or by clicking the run button in your IDE.
7. **Test the application:** Once the application is built and running, you can test the different features to ensure that everything is working correctly.

These are the general steps to run a Flutter project that uses Firebase. Keep in mind that some steps may vary depending on your project's specific requirements.

CHAPTER VII

CONCLUSION & FUTURE SCOPE

7.1 Conclusion:

In conclusion, the student-based application we are creating is a comprehensive platform for students to ask and answer questions, share knowledge and resources, and access study materials. With this application, students can easily collaborate with their peers, stay updated with the latest news and announcements, and enhance their learning experience.

The application provides several modules such as asking questions and providing answers, uploading posts and materials in different formats, and a material section with various fields. The home page displays all the uploaded posts, making it easy for students to stay updated with the latest news and announcements.

The application's main goal is to facilitate learning and help students achieve their academic goals. By providing a platform for students to ask and answer questions, share resources, and access study materials, the application helps students enhance their knowledge and understanding.

7.2 Future Scope:

There are several future scope options that we can explore to make the application more user-friendly and effective. These include:

1. **Personalization:** Adding a personalization feature that allows students to customize the application according to their preferences and needs can enhance the user experience.
2. **Collaboration:** Introducing a collaboration feature that allows students to work on projects and assignments together can promote teamwork and learning.
3. **AI-powered features:** Incorporating AI-powered features such as chatbots and recommendation engines can provide personalized learning experiences to students.
4. **Analytics:** Adding an analytics feature that allows teachers to track students' progress and identify areas that need improvement can enhance the effectiveness of the application.

5. **Gamification:** Introducing a gamification element in the application, such as rewards and points, can make learning more engaging and fun for students.
6. **Integration with other platforms:** Integrating the application with other platforms such as social media and messaging apps can make it more accessible and user-friendly.

Overall, by incorporating these future scope options, we can enhance the functionality of the application and make it more effective in helping students achieve their academic goals. The application has the potential to become a comprehensive learning platform that benefits students in their academic pursuits.

CHAPTER VIII

REFERENCES

1. Zoe Karabatzaki, Agathi Stathopoulou, Georgia Kokkalia “Mobile Application Tools for Students in Secondary Education. An Evaluation Study.”, International Journal of Interactive Mobile Technologies (iJIM) 12(2):142, March 2018.
2. Flutter: <https://flutter.dev/>
3. Firebase: <https://firebase.google.com/>
4. Firebase full course by freeCodeCamp.org:
https://www.youtube.com/watch?v=fgdpvwEWJ9M&list=PLute5tpjW6-g_FPXKnHvTgC2py2J0Gseg&index=25&t=524s
5. Firebase for Flutter: <https://firebase.flutter.dev/>
6. Flutter and Firebase: Build a Complete App for iOS and Android:
<https://www.udemy.com/course/flutter-firebase-build-a-complete-app-for-ios-android/>
7. Instagram clone for reference:
https://www.youtube.com/watch?v=mEPm9w5QIJM&list=PLute5tpjW6-g_FPXKnHvTgC2py2J0Gseg&index=23
8. Building a CRUD application using Flutter and Firebase:
<https://medium.com/flutter-community/building-a-crud-application-using-flutter-and-firebase-249efd8e7a34>
9. Firebase Setup docs:
<https://firebase.google.com/docs/flutter/setup?platform=android>
10. Firebase Authentication with Flutter:
<https://firebase.google.com/docs/auth/flutter/start>
11. LinkedIn for Reference: <https://www.linkedin.com/>