Alliance

PROJECT SUPERVISOR

Ms. Sobia Iftikhar

PROJECT CO-SUPERVISOR

Mr. Shoaib Raza

PROJECT TEAM

Sarim Sohail	K190153
Mubashira Khan	K190239
Hassan Jamil	K191292

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

May 2023

Project Supervisor	Ms. Sobia Iftikhar	
Project Team	Sarim Sohail	K19-0153
	Mubashira Khan	K19-0239
	Hassan Jamil	K19-1292
Submission Date	May 2, 2023	

Ms. Sobia Iftikhar _____

Supervisor

Mr. Shoaib Raza _____

Co-Supervisor

Dr. Zulfiqar Ali Memon _____

Head of Department

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

Acknowledgement

We are deeply grateful to the almighty Allah for granting us the strength, knowledge, and perseverance to complete this mobile app project. Without His guidance and blessings, this project would not have been possible.

We would like to extend our heartfelt gratitude to our supervisor Ms. Sobia Iftikhar for her continuous guidance, support, and encouragement throughout the project. Her valuable insights and constructive feedback were crucial in shaping the direction of the project.

We are also grateful to our co-supervisor Mr. Shoaib Raza for his assistance in providing valuable suggestions and insights in the development of the mobile app. His expertise and experience were instrumental in ensuring the project's success.

Finally, we would like to express our appreciation to all those who have supported us throughout this journey, including our family and friends. Their unwavering support and encouragement were essential in keeping us motivated and focused throughout the project.

Abstract

Alliance is a comprehensive mobile application designed to connect students in Karachi with qualified tutors for academic support. Developed using the Flutter framework, the application provides an intuitive interface that enables students to search for tutors based on subject, location, and availability. The app's features include in-app messaging, video conferencing, and a contract system to facilitate tutor-student interactions. With a focus on personalized learning, Alliance empowers students to achieve their academic goals by providing access to quality tutoring services from the comfort of their own homes. The platform also supports tutor recruitment and onboarding, ensuring a high standard of teaching and learning outcomes. Overall, Alliance represents an innovative solution to the challenges faced by students in accessing quality education, particularly in the context of remote learning.

Contents

This page is intentionally left blank.

1. Introduction

1.1. Overview

The necessity for a tutoring service in Karachi is the basis for this application. There are many knowledgeable people and students in Karachi who can mentor and instruct those who are younger than themselves, but there is no dedicated platform built for this kind of connection. Apart from well-known coaching facilities, another challenge that home tutors frequently encounter is that of word-of-mouth advertising. The target users of this application will be private or home tutors and students looking for these services. It will be a fully working application that is accessible on any android device. In order to facilitate this kind of engagement and make tutoring easily accessible to Karachi residents, we seek to offer a comprehensive and coherent platform. Through this project, we hope to expand the amount of educational volunteer work being done in underprivileged areas and to offer a secure platform for interactions regarding tutoring.

1.2. Motivation

There are many knowledgeable people and students in Karachi who can mentor and instruct those who are younger than themselves, but there is no dedicated platform built for this kind of connection. Apart from well-known coaching facilities, another challenge that home tutors frequently encounter is that of word-of-mouth advertising. The target users of this application will be private or home tutors and students looking for these services. It will be a fully working application that is accessible on any android device. Through this project, we hope to expand the amount of educational volunteer work being done in underserved communities and to offer a secure venue for conversations about tutoring.

1.3. Problem Statement

Within our city there are many qualified individuals and students that are capable of educating those younger than them in various subjects, however there is a lack of a cohesive and dedicated platform built to service this kind of interaction.

1.4. Objectives

Following is the list of the objectives that will address the identified problem:

- Provide education to the underprivileged
- Assist students & tutor to communicate with each other
- Provide opportunity to students capable of providing education to those less experienced than them
- Provide job opportunities to qualified individuals
- Allow students to find tutors from the comfort of their homes
- Attend online classes
- Allow a student to ‘request’ a tutor meeting their requirements
- allow users to easily schedule classes and set reminders ahead of time

2. Related Work

2.1. Comparison Chart

This chart provides the comparison between Alliance application and the previous similar applications.

Figure 1: Comparison Chart

2.2. Literature Review

Tutors-A is a tutoring app that offers users the ability to chat with tutors, providing a convenient platform for communication. However, the app is not without its shortcomings. One notable drawback is the presence of errors, which can hinder the smooth user experience. These errors may arise during various interactions within the app, potentially causing frustration and inconvenience for both students and tutors. Additionally, while *Tutors-A* boasts extensive input details, allowing users to provide comprehensive information, this can sometimes result in overwhelming forms or excessive data entry requirements. The app could benefit from streamlining the input process to ensure a more user-friendly experience.

Tutors-B is a tutoring app that aims to connect students and tutors; however, the app is plagued with numerous issues and limitations. One of the significant drawbacks of *Tutors-B* is its error-prone registration process. Users often encounter errors and difficulties when attempting to create an account, resulting in frustration and hindrances in accessing the app's features. Moreover, *Tutors-B* is riddled with bugs and glitches that significantly impact its functionality. These technical flaws lead to a subpar user experience, impeding smooth navigation and interactions within the app. Due to these persistent issues, *Tutors-B* falls short in delivering a reliable and efficient platform for tutoring services.

Abwaab is a tutoring app that offers basic features such as profile editing, user registration, tutor exploration, and the ability to become a tutor. While these features provide some functionality within the app, *Abwaab* falls short in several aspects. The profile editing feature, while present, lacks robust customization options, limiting users' ability to showcase their expertise effectively. The registration process, although available, may not be as seamless and user-friendly as desired, potentially leading to confusion or complications during account creation. Furthermore, the tutor exploration feature in *Abwaab* is relatively limited compared to other tutoring apps, making it more challenging for students to find the most suitable tutors for their specific needs. Additionally, *Abwaab* offers premium features that are locked behind a paywall, which restricts access to enhanced functionalities and may limit the overall user experience for those who are unable or unwilling to subscribe. Due to these limitations and restrictions, *Abwaab* may not provide the comprehensive and user-centric tutoring experience that users seek.

Munzil is a tutoring app that aims to connect students with tutors, but it is marred by a notable issue – errors. Users frequently encounter errors while using *Munzil*, which can disrupt the app's functionality and hinder a seamless user experience. These errors may manifest during various interactions, such as profile editing, searching for tutors, or accessing other features within the app. The presence of these errors can lead to frustration and inconvenience for both students and tutors, potentially impacting their ability to effectively utilize the platform for learning purposes. As a result, *Munzil's* reliability and overall usability may be compromised, making it a less desirable choice compared to other tutoring apps that offer smoother and error-free experiences.

Alliance is a cutting-edge tutoring app that excels in providing a comprehensive set of features to facilitate seamless learning experiences for students and tutors alike. This app stands out for its remarkable ability to offer a wide range of functionalities without the presence of errors, ensuring a smooth and hassle-free user experience. Students and tutors can leverage Alliance's intuitive interface and reliable infrastructure to enjoy uninterrupted interactions and efficient communication.

Alliance encompasses essential features such as chat functionality, contract management, virtual meeting capabilities, profile editing, user registration, volunteer requests, tutor exploration, and the ability to become a tutor. These features empower users to engage in meaningful educational discussions, conveniently manage tutoring

agreements, schedule and participate in virtual meetings, and fine-tune their profiles to showcase their expertise effectively. The user registration process is seamless and straightforward, allowing individuals to quickly create their accounts and start benefiting from the app's extensive offerings.

Furthermore, *Alliance* places a strong emphasis on tutor exploration, enabling students to find the perfect match for their learning needs. The app offers comprehensive search filters based on subjects, qualifications, and ratings, ensuring that students can make informed decisions when selecting tutors.

While *Alliance* does not provide premium features or extensive input options, these limitations do not detract from its overall excellence. Instead, the app focuses on delivering a streamlined and reliable tutoring experience, ensuring that users can engage in productive learning sessions without distractions or unnecessary complexities.

In summary, *Alliance* sets a new standard for tutoring apps by offering a wide array of features, impeccable functionality, and a user-friendly interface. With its emphasis on error-free performance and essential features, *Alliance* is an app that truly revolutionizes the way students and tutors connect and learn together.

3. Analysis

3.1. Project Scope

The application will target users that need tutoring services. The user can be either a student or tutor. Users will provide necessary details to log in or register themselves on the application. A user may set his/her location, look for tutors around him/her, may request to get a tutor, arrange a video call session with another user, view their personal profile and view their scheduled sessions.

3.2. Not in Scope

There is no intention to make this app cross-platform.

3.3. Stakeholders

3.3.1. Internal Stakeholders

- Product owner
- Product developer

3.3.2. External Stakeholders

- Service consumers – student, parents, tutor

- Service providers - bank

3.4. Operating Environment

Alliance is an Android based Mobile application requiring the latest android OS.

3.5. System Constraints

3.5.1. Software Constraints

- System execution on Visual Studio Code.
- Android studio must be installed.
- Firebase database
- Android Device Manager
- scrcpy
- Flutter
- javasdk

3.5.2. Hardware Constraints

Android mobile running the latest android OS.

3.5.3. User Constraints

- Users must have a valid email address to register themselves.
- Users must provide authentic documents when asked to provide academic qualifications.
- Users must pass the test to qualify as a tutor.

3.6. Assumptions and Dependencies

3.6.1. Assumptions

User has a valid email address.

3.6.2. Dependencies

- Firebase [NoSQL database] – lets store & sync users in real time.
- Android device with latest android OS
- Third-party payment getaway for online payment transactions.
- Third-party API for video call sessions.

3.7. Technology Stack

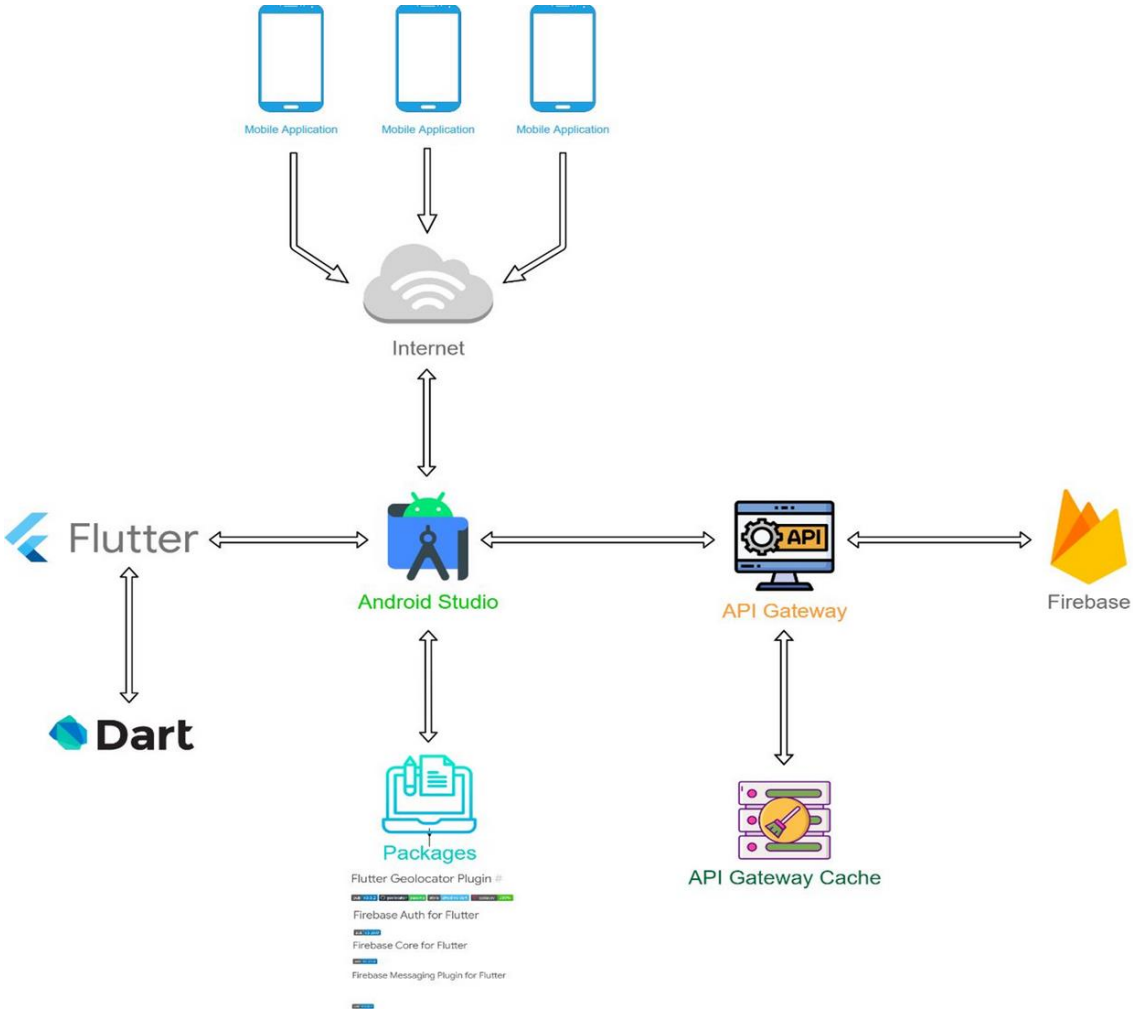

Figure 2: Technology Stack

4. Requirements

1. External Interface Requirements

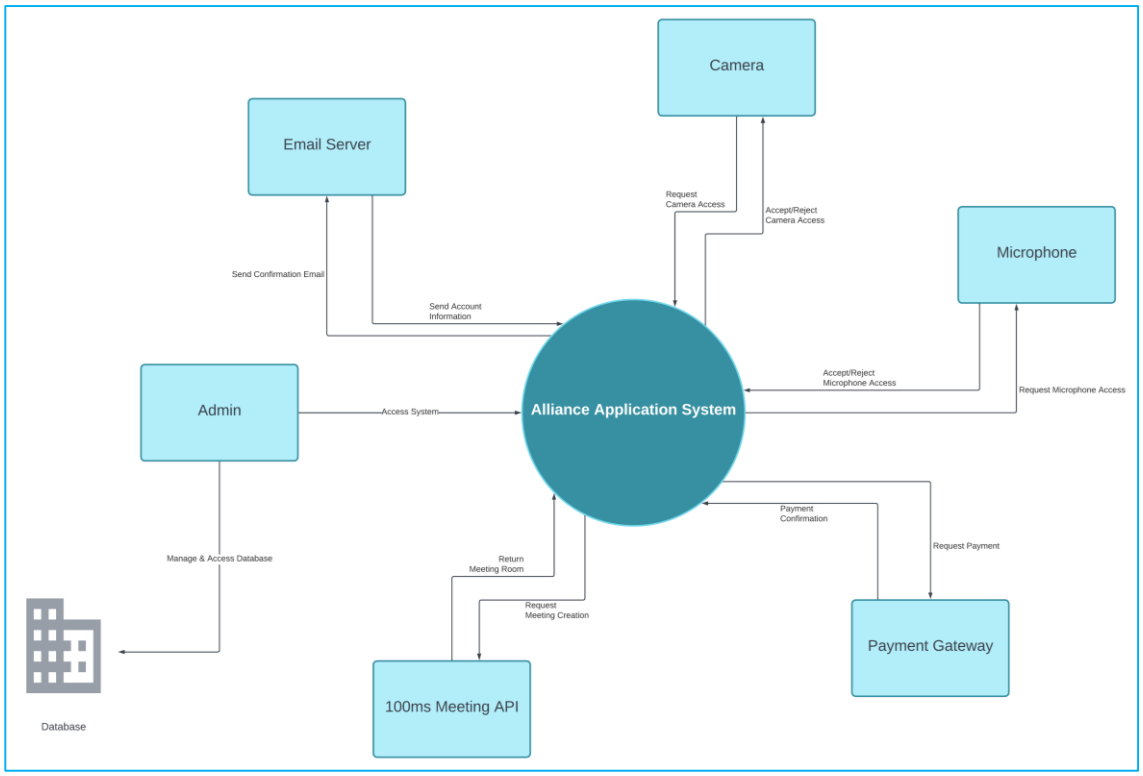

Figure 3: Context Diagram

4.1.1. Hardware Interfaces

The application uses APIs to send and get data from the server. Any device executing the latest Android OS can install the application.

4.1.2. Software Interfaces

Flutter [Dart] has been used to build the front end & back end of the application. Firebase, a NoSQL database, has been used to store data of the users. The application uses a third-party payment gateway for online transactions. To provide location access, Google Maps has been used. Application uses the 100ms third-party API gateway for meetings. Camera is used to attend video call meetings. A microphone is used for audio input in video call meetings.

4.1.3. Communication Interfaces

The project will contain a real-time video meeting like zoom, with audio and video functionality, camera switching and a built-in chat for the meeting itself. This will be achieved through API integration of our application with the HMS API. Furthermore, real time chat functionality will be added, achieved using a NOSQL database, alongside user verification as well.

4.2. Functional Hierarchy

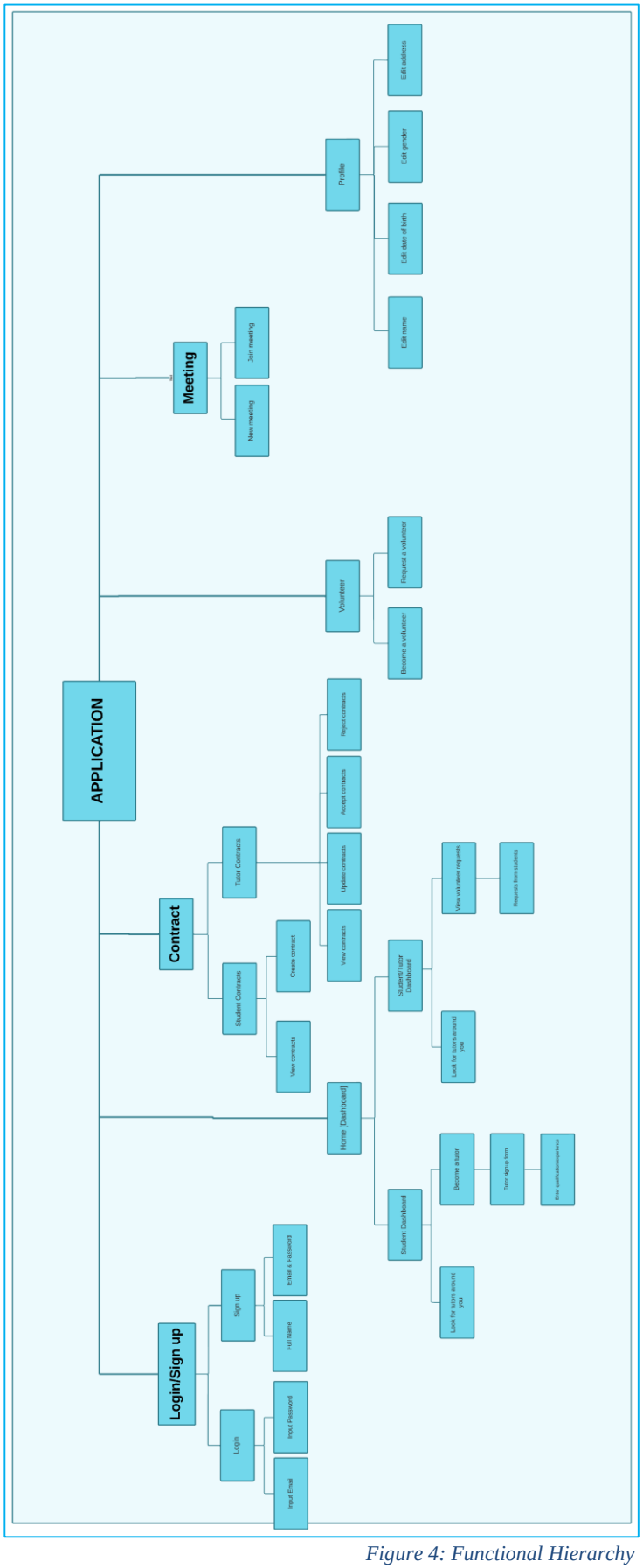

Figure 4: Functional Hierarchy

4.3. Use Cases

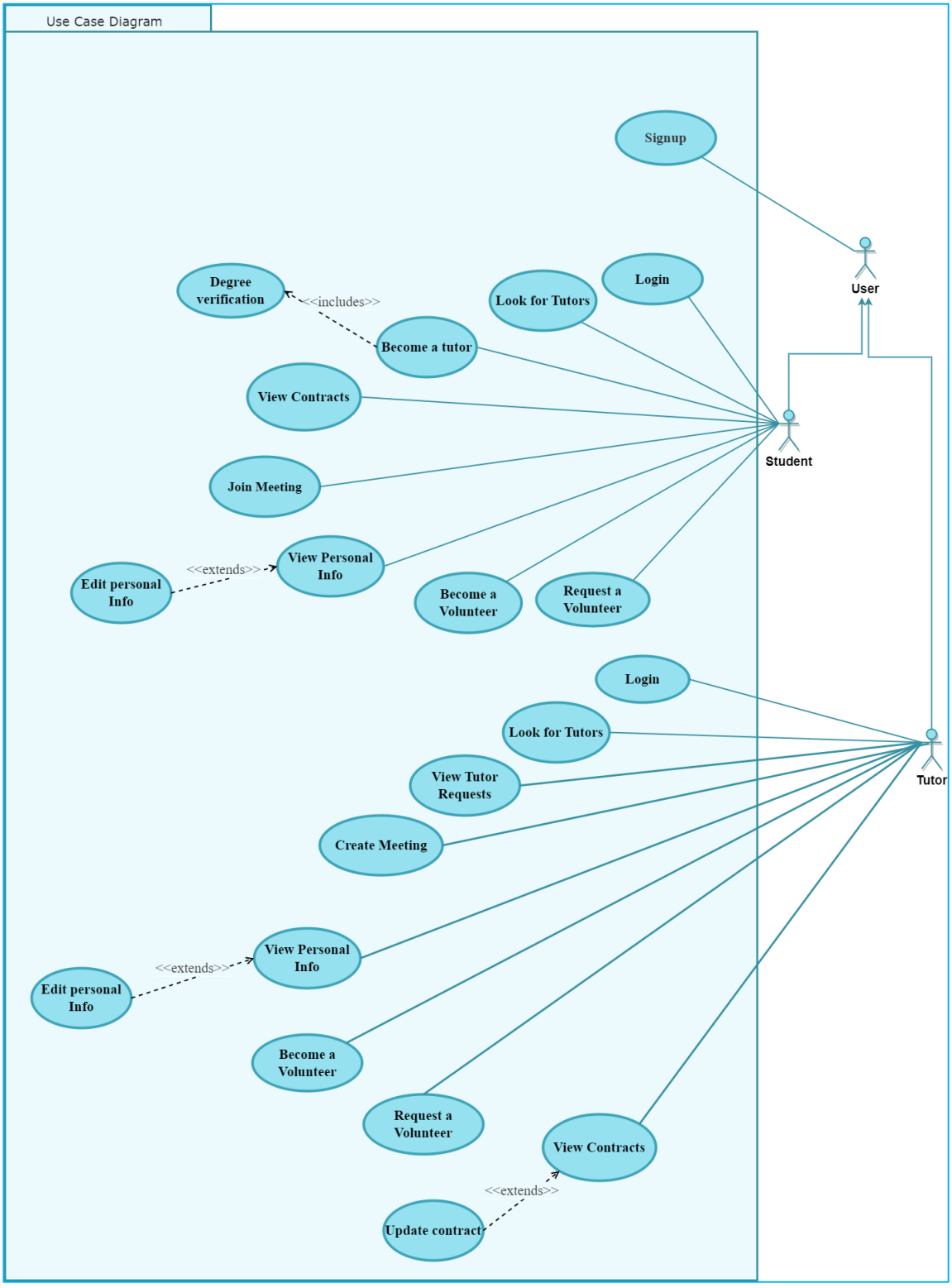

Figure 5: Use Case Diagram

<UC 1: Signup/Register>

Use case Id:	UC 1
Actors:	User [Student & Tutor]
Feature:	When user clicks on the register button.
Pre-condition:	o Stable internet connection. o User is not registered.

Scenarios

Step#	Action	Software Reaction
1.	Press Register button on Login screen.	Registration screen is displayed.
2.	Input full name, email address, date of birth, location, and password.	
3.	Press register button to start process.	o Validate user input. o Create user account. o User data stored on database.

Alternate Scenarios: Invalid user input in registration form.

1a. System identifies invalid user input.
1b. System displays error to user.
1c. User re-enters information.

Post Conditions

Step#	Description
1.	User data stored in database.
2.	User can now log in to system.

Use Case Cross referenced	-

<UC 2: Login>

Use case Id:	UC 2
Actors:	User [Student & Tutor]
Feature:	Displayed right after Welcome screen.
Pre-condition:	○ Stable internet connection. ○ User is a registered-on system ○ User has credentials to log in to system.

Scenarios

Step#	Action	Software Reaction
1.	User opens the app.	Log in screen displayed right after Welcome screen.
2.	Input email address.	
3.	Input password.	
4.	User presses log in button.	System validates credentials from database. Upon validation, user has logged in successfully.

Alternate Scenarios: Invalid user input.

1a. System is unable to match the given credentials in the database.
1b. System displays error to user.
1c. User re-enters information.

Post Conditions

Step#	Description
1.	User has successfully logged in to system.

Use Case Cross referenced	Signup

<UC 3: Become a tutor>

Use case Id:	UC 3
Actors:	Student
Feature:	Displayed on Student dashboard.
Pre-condition:	○ Stable internet connection. ○ User is currently a student.

Scenarios

Step#	Action	Software Reaction
1.	User views dashboard.	
2.	User presses Become a Tutor option.	Leads user to a tutor signup form.
3.	User inputs academic qualification, experience, mode of tutoring, degree verification and their schedule	
4.	User waits for system to verify their information.	Logs the tutor out to resign again

Alternate Scenarios: **1.** User inputs incorrect information.
2. User fails the degree validation.

1a. System displays error to user.
1b. User re-enters information.

2a. User can reupload and re-verify degree in their profile section for verification.

Post Conditions

Step#	Description
1.	User passes the test.
2.	User is now a tutor.

Use Case Cross referenced	Clear tests.

<UC 4: Request a Volunteer>

Use case Id:	UC 4
Actors:	User [Student & Tutor]
Feature:	Displayed on dashboard.
Pre-condition:	o Stable internet connection. o User is a student (can be a tutor).

Scenarios

Step#	Action	Software Reaction
1.	User presses the Request a Tutor option.	Leads user to Request Tutor form.
2.	User inputs price range, desired timings, desired qualification for tutor, and location.	
3.	Users can optionally enter any special requests.	
4.	User presses Submit.	Screen confirms that form is submitted.

Alternate Scenarios: unstable internet connection.

1a. Progress is lost.
1b. User re-fills the form.

Post Conditions

Step#	Description
1.	User has successfully posted tutor request.

Use Case Cross referenced	Log in

<UC 5: View Contracts>

Use case Id:	UC 5
Actors:	User [Student & Tutor]
Feature:	Contract section on main dashboard.
Pre-condition:	○ Stable internet connection. ○ User wants to view the scheduled sessions for him/her.

Scenarios

Step#	Action	Software Reaction
1.	User presses Contracts from the nav bar.	Leads user Contracts screen.
2.	The user can view their current contracts, if they're a tutor they can view the contracts they're in charge of.	Contracts are displayed in their respective sections based on user being a student or a tutor.

Alternate Scenarios: -2. User has no contracts

Post Conditions 2a. User sees that there are no pending contracts available

Step#	Description
1.	User can view their contracts.

Use Case Cross referenced	Log in

<UC 6: Join Meeting>		
Use case Id:	UC 6	
Actors: User [Student & Tutor]		
Feature: Meeting section on main dashboard.		
Pre-condition:	o Stable internet connection. o User wants to join a meeting.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User presses Meeting from the nav bar.	Leads user Meeting screen.
2.	User presses the Join Meeting button.	Pop up to enter meeting link.
3.	User inputs meeting link and presses Join button.	
4.	User enters the meeting.	
Alternate Scenarios: Invalid meeting link		
1a. User is asked to re-enter the link.		
Post Conditions		
Step#	**Description**	
1.	User has joined the meeting successfully.	
Use Case Cross referenced	Log in	

<UC 7: Create Meeting>

Use case Id:	UC 7
Actors:	User [Student & Tutor]
Feature:	Meeting section on main dashboard.
Pre-condition:	○ Stable internet connection. ○ User [tutor] wants to create meeting to conduct a class.

Scenarios

Step#	Action	Software Reaction
1.	User presses Meeting from the nav bar.	Leads user Meeting screen.
2.	User presses the New Meeting button.	System generates a meeting link which is returned to tutor.
3.	User can use the system generated link to connect with students.	

Alternate Scenarios: If link not generated in 10 seconds.

1a. User is returned to the Meeting screen.
1b. User then repeats the same process, if user wishes to create a new meeting session.

Post Conditions

Step#	Description
1.	User has created meeting successfully.

Use Case Cross referenced	Log in, Join meeting

<UC 8: Accept Contract>		
Use case Id:	UC 8	
Actors: Tutor		
Feature: Contract section on main dashboard.		
Pre-condition:	○ Stable internet connection. ○ User [tutor] has received a contract offer	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User presses Contract from the nav bar.	Leads user Contract screen.
2.	User [tutor] views any contracts offered in the pending contract section.	
3.	User [tutor] can accept or deny the contract.	System updates both the sent users and accepting users contracts as accepted or denied, respectively.
Alternate Scenarios: - 1. User denies contract		
Post Conditions 1a. The contract and it's details are deleted from the user after confirmation of deletion.		
Step#	**Description**	
1.	User has accepted contracts successfully.	
Use Case Cross referenced	Log in	

<UC 9: View Personal Info>		
Use case Id:	UC 9	
Actors: User [Student & Tutor]		
Feature: Profile section on main dashboard.		
Pre-condition:	o Stable internet connection. o User wants to view their profile information.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User presses Profile from the nav bar.	Leads user Profile screen.
2.	User views personal information such as name and email address.	
Alternate Scenarios: -		
Post Conditions		
Step#	**Description**	
1.	User can view Profile.	
Use Case Cross referenced	Log in, Edit Personal Info	

<UC 10: Edit Personal Info>		
Use case Id:	UC 10	
Actors: User [Student & Tutor]		
Feature: Profile section on main dashboard.		
Pre-condition:	○ Stable internet connection. ○ User wants to edit their profile information.	
Scenarios		
Step#	**Action**	**Software Reaction**
1.	User presses Profile from the nav bar.	Leads user Profile screen.
2.	User can edit name, email address and other personal details.	System updates the user information in the database.
3.	User can view the personal information after editing.	
Alternate Scenarios: Unstable internet connection		
1a. User information not updated on screen. **1b.** No changes to user information in database.		
Post Conditions		
Step#	**Description**	
1.	User can view and edit Profile.	
Use Case Cross referenced	Log in, View Personal Info	

4.4. Non-functional Requirements

4.4.1. Performance Requirements

- The application will load in a few seconds depending on the speed of the internet connection.
- The precision of the application has been kept best to developers' abilities.
- The application shall facilitate multiple users at a time.
- The application's capacity shall depend upon the hardware components of the user's device.
- The application takes measures to ensure user's safety in terms of data protection.
- The application's reliability will be predicted as it goes through the testing phases.

4.4.2. Safety Requirements

The system has been designed to avoid any possible loss, damage, or harm.

4.4.3. Security Requirements

- The application uses a signup/login system that requires user's personal information.
- The application uses a payment system that may require user's confidential information for online transactions.
- The system applies adequate constraints on the database to secure user data.
- The application shall automatically log out all users after a certain period of account inactivity or software update.

4.4.4. User Documentation

The system is designed in a way which makes it self-explanatory and easy to use hence there will not be any need for user documentation. It also has an FAQ section to assist users about any queries they may have about the application.

5. Design

5.1. System Architecture

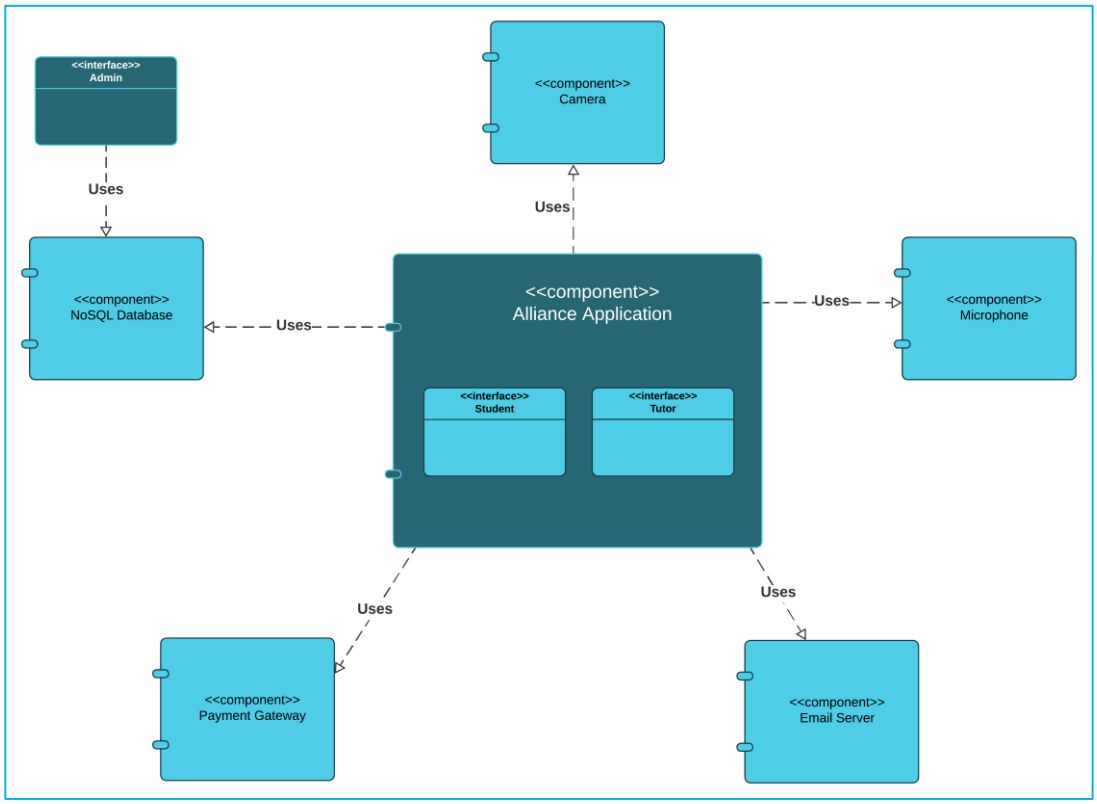

Figure 6: System Architecture

5.2. Software Architecture

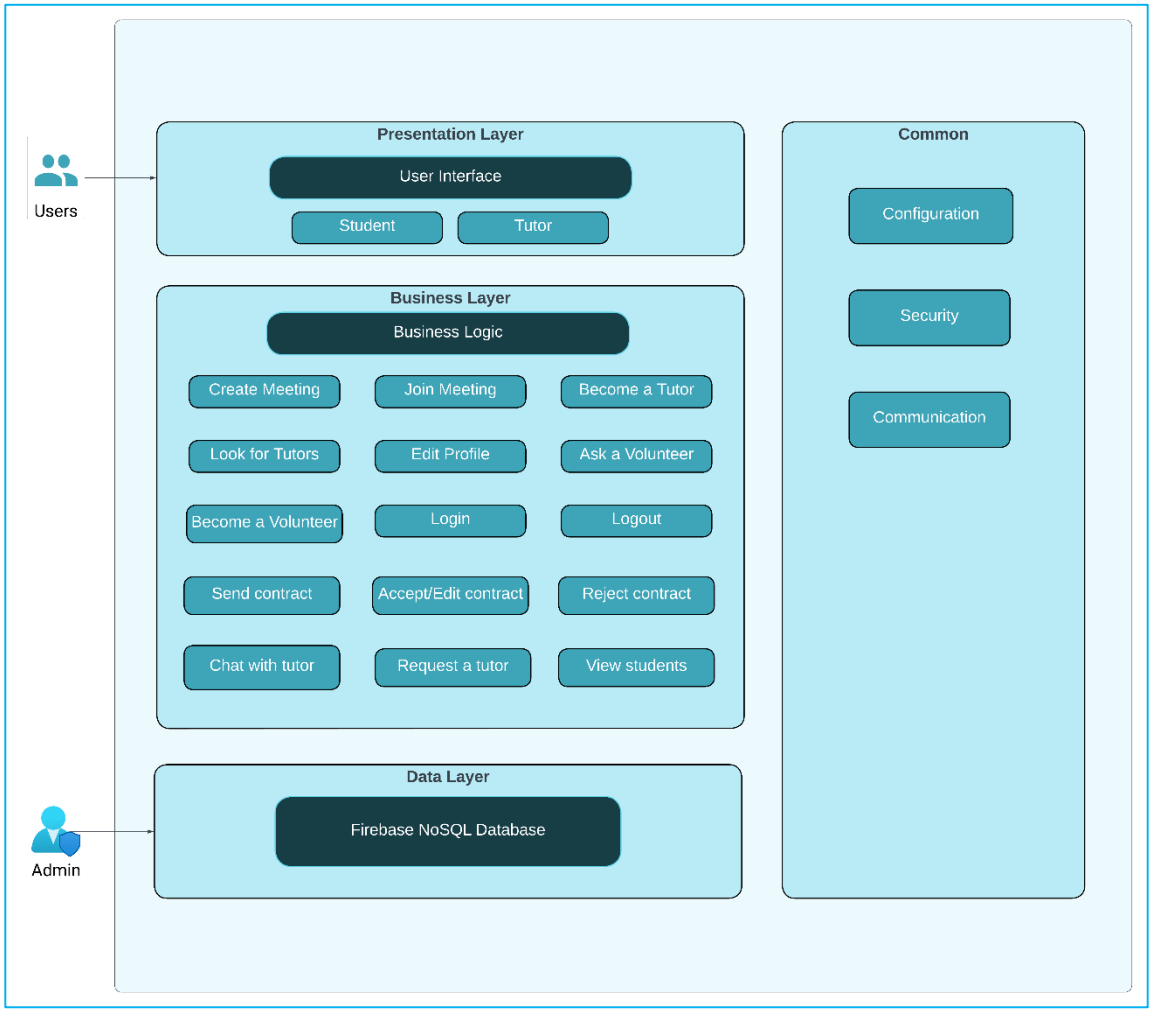

Figure 7: Software Architecture

5.3. Design Strategy

- The applications frontend is implemented using Flutter's frontend design widgets. Flutters design widgets are reusable components that ensure design consistency across the entire application, and it also makes the codebase to run efficiently and smoothly.
- The applications backend is also implemented using Flutter's object-oriented approach. The object-oriented approach of Flutter ensures that the design of the system meets all the important requirements.
- For the real time chat feature and all other information storage of the users, the NoSQL cloud database Firebase is used.
- For the online transactions performed through the application a third-party payment gateway is used.
- The system is designed in a modular manner to make it simple to add new components in the future. Only one module needs to be altered if any changes are made; the rest of the system won't be impacted.
- For the online classes and meetings, a third-party API is used.

5.4. Detailed System Design

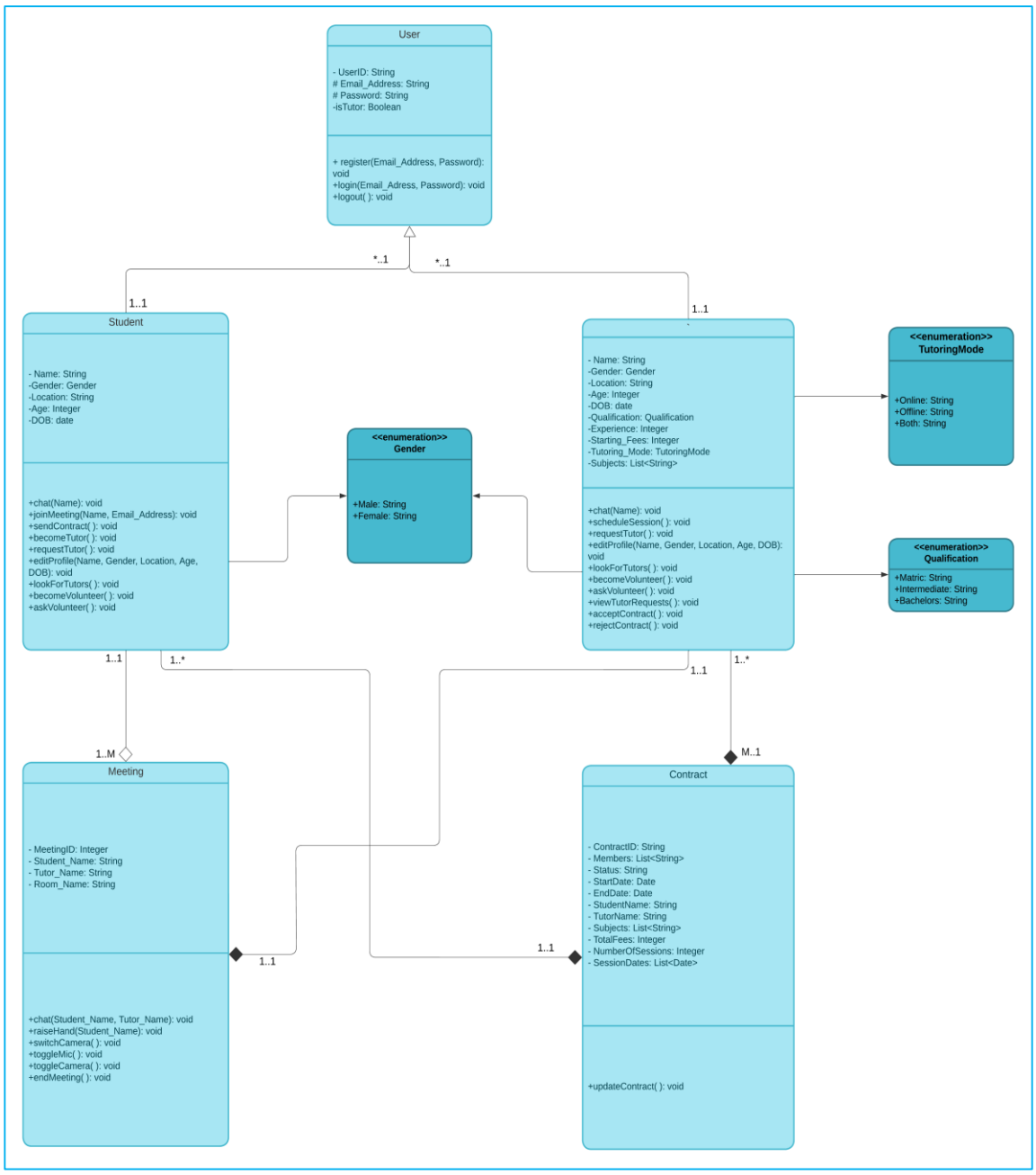

Figure 8: Class Diagram

5.4.1. ER Diagram

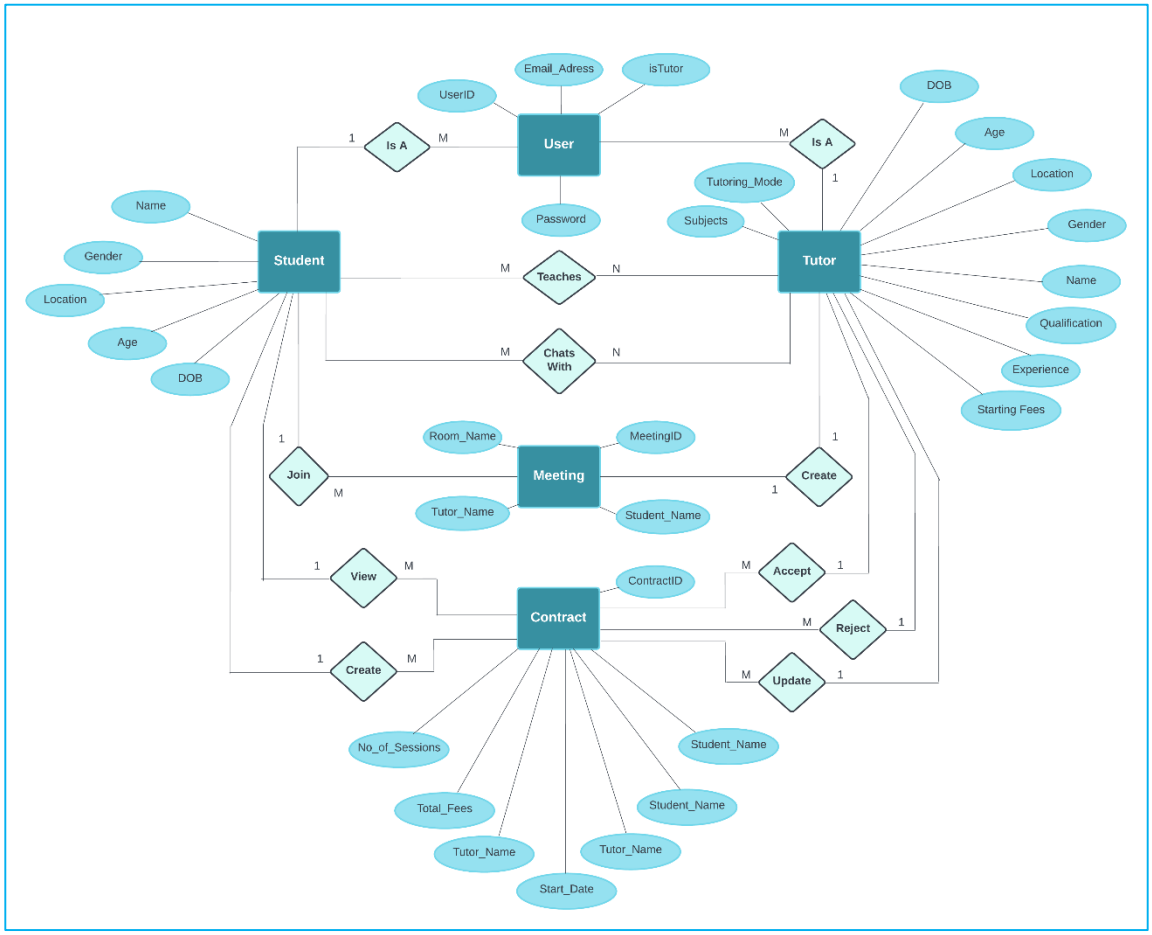

Figure 9: ERD

5.4.2. Data Dictionary

I. User

User						
Name	User					
Alias	-					
Where-used/how-used	○ User can either be a student or a Tutor ○ User can register themselves into the application ○ Many users can be students ○ Many users can be tutors Many users can be a Student and a Tutor at the same time					
Content description	Notation for representing content.					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
Email_Add	The email	String	-	No	-	PK

ress	address that user uses to register into the application and that uniquely identifies a user.					
isTutor	A Boolean variable indicating whether a user is a tutor or not.	Boolean	1	No	False	
Name	The full name of the user.	String	30	No	-	-
Gender	A value indicating the gender of the user	String	10	No	-	-
Address	The selected or specified area wise address of the student.	String	50	No	-	-
DOB	The student's date of birth.	Date	-	-	-	-
Conversations	The list of email ID's of the users, the user has a conversation with	List <String>	-	Yes	-	-
Profile Picture	The link of the users profile picture stored in database	String	-	Yes	" "	-

II. Student

Student	
Name	Student
Alias	-
Where-used/how-used	○ Student is a user. ○ Student can be taught by many Tutors. ○ Students can chat with many Tutors. ○ Student can join a Meeting. ○ Student can create/send contracts to many Tutors. ○ Student can view their pending and ongoing contracts.
Content description	Notation for representing content.

Column Name	Description	Type	Length	Null able	Default Value	Key Type
StudentID	A unique identification number assigned to each student.	String	4	No	-	-
Email Address	The email address that the student uses to register into the application and that uniquely identifies a student.	String	-	No	-	PK
Name	The full name of the student.	String	30	No	-	-
Gender	A value indicating the gender of the student	String	10	No	-	-
Address	The selected or specified area wise address of the student.	String	50	No	-	-
DOB	The student's date of birth.	Date	-	-	-	-
isTutor	A Boolean variable indicating whether the student is a tutor or not.	Boolean	1	No	False	
Conversations	The list of email ID's of the users, the student has a conversation with	List <String>	-	Yes	-	-
Profile Picture	The link of the users profile picture stored in database	String	-	Yes	" "	-

III. Tutor

Tutor	
Name	Tutor
Alias	-
Where-used/how-used	○ The tutor is a user. ○ Tutor can teach many students ○ Tutor can chat with many students ○ Tutor can send/create many contracts ○ Tutor can accept/reject many contracts ○ Tutor can view pending/ongoing contracts ○ Tutor can updated ongoing contracts ○ Tutor can create a meeting
Content description	Notation for representing content.

Column Name	Description	Type	Length	Null able	Default Value	Key Type
TutorID	A unique identification number assigned to each student.	String	4	No	-	-
Email Address	The email address that the tutor uses to register into the application and that uniquely identifies a tutor.	String	-	No	-	PK
Name	The full name of the tutor.	String	30	No	-	-
Gender	A value indicating the gender of the tutor.	String	10	No	-	-
Address	The selected or specified area wise address of the tutor.	String	50	No	-	-
DOB	The tutor's date of birth.	Date	-	No	-	-
Profile Picture	The link of the tutor's profile picture stored in database	String	-	Yes	" "	-
Qualification	The tutors specified qualification	String	-	No	-	-

	level.					
Experience	The total teaching experience of the tutor in years.	Integer	-	No	-	-
Current Employment	The current employment the tutor has.	String	-	Yes	-	-
Degree Number	The degree number of the tutors degree.	String	8	No	-	-
Degree Document	The link to the tutors degree document the he/she scans and uploads	String	-	No	-	-
inContract	The Boolean variable that indicates whether the student has any ongoing contracts or not	Boolean	5	No	False	-
isVerified	The Boolean variable the indicates whether the tutor is a verified tutor or not	Boolean	5	No	False	-
Number of Reviews	The total number of reviews that the tutor gets from students	Integer	-	Yes	-	-
Rating	The calculated rating of the tutor based on the number of reviews	Double	-	Yes	0	-
Subject Prices	The list of subjects the tutor selects and their specified prices that the tutor inputs.	List < Dictionary>	-	No	-	-
Subject Timings	The list of subjects the tutor selects	List <Dictionary >	-	No	-	-

	and their specified timings that the tutor selects.					
Days of Week	The list of days of week the tutor selects to teach on.	List <String>	-	No	-	-
Tutoring Mode	The mode of tutoring the tutor selects to teach on	String	-	No	-	-

IV. Meeting

Meeting						
Name	Meeting					
Alias	-					
Where-used/how-used	○ A meeting can be created by a Tutor ○ A meeting can be joined by many students					
Content description	Notation for representing content.					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
MeetingID	A unique identification number assigned to each meeting.	String	3	No	-	PK
Tutor_Name	The name of the tutor that created the meeting.	String	-	No	-	-
Student_Name(s)	The list of the names of the students that joined the meeting	List <String>	-	No	-	-

V. Contract

Contract	
Name	Contract

Alias	-
Where-used/how-used	○ Many contracts can be sent/created by a Student/Tutor ○ A Student can view many contracts ○ A Tutor can accept/reject many contracts ○ A Tutor can update many ongoing contracts
Content description	Notation for representing content.

Column Name	Description	Type	Length	Null able	Default Value	Key Type
ContractID	A unique identification number assigned to each contract.	String	-	No	-	PK
Student_Name	The name of the student in the contract	String	-	No	-	-
Tutor_Name	The name of the tutor in the contract	String	-	No	-	-
Members	A list of the email IDs of the student and the tutor	List <String>	-	No	-	-
Start_Date	The starting date of the contract	Date	-	No	-	-
End_Date	The date on which the contract will end	Date	-	No	-	-
Status	The status of the contract which indicates whether the contract is pending or ongoing	String	-	No	-	-
Subjects	The list of subjects the tutor is teaching in the contract	List<String>	-	No	-	-
Days	The list of days on which the tutor will be teaching	List<String>	-	No	-	-
Number of Sessions	The total number of	Integer	-	No	-	-

	sessions that will be taught during the contract					
Session Dates	The list of dates on which the tutor will be teaching and each dates' comments	List<Dictionary>	-	No	-	-
Total Fees	The total fees of the contract	Integer	-	No	-	-

6. Implementation

6.1. Sequence Diagram

6.1.1. Registration

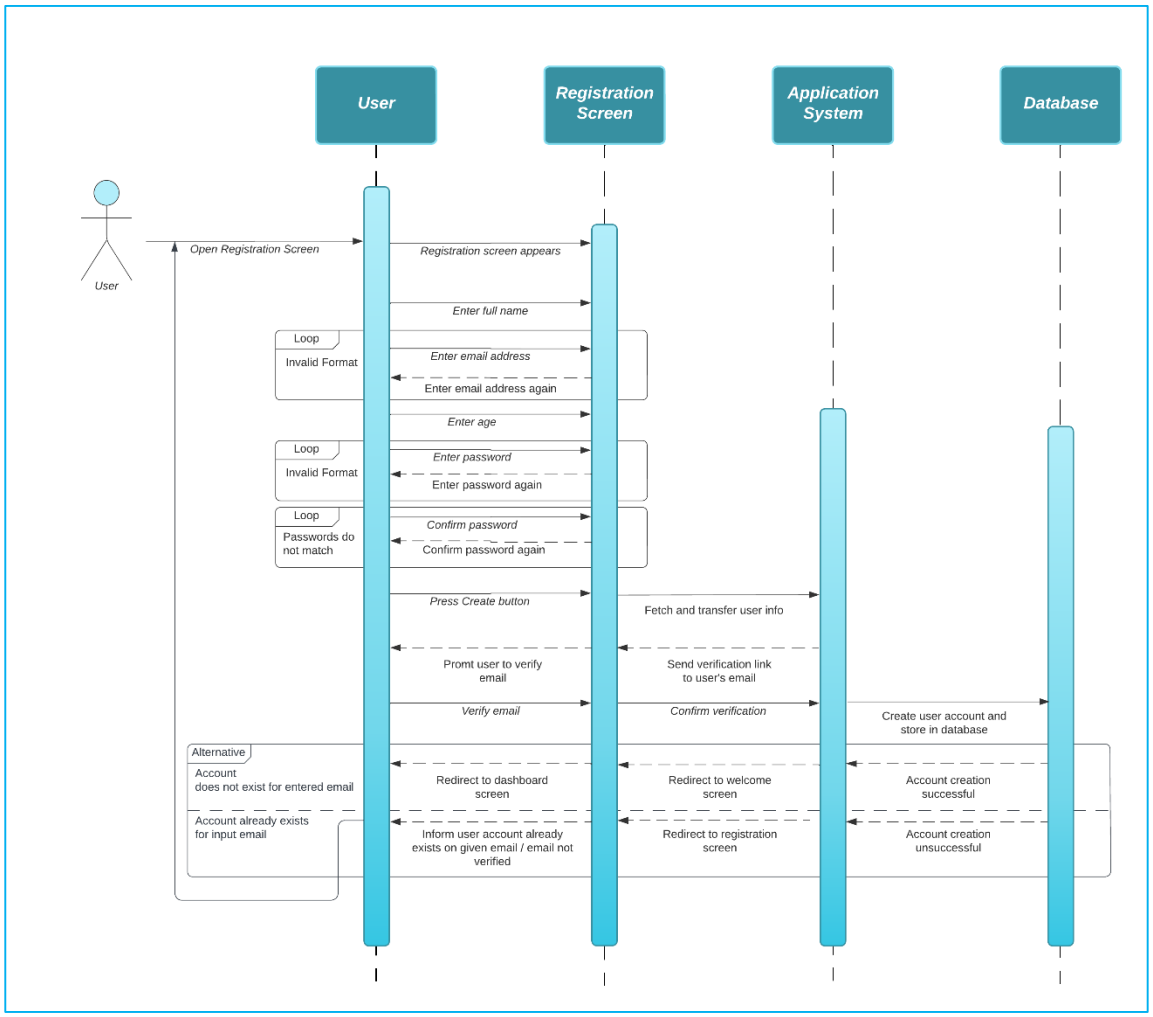

Figure 10: Registration Sequence Diagram

1. User opens the Registration screen.
2. Registration screen appears.
3. User enters their full name.
4. User enters their email address.
5. If the format for email address is invalid, user enters email address again.
6. User enters password.
7. If the requirement is not met for the password, user enters password again.
8. User presses Create button.
9. User data is fetched from the registration screen and transferred to the application system.
10. A verification link is sent to the user's email address by the application.

11. The user is prompted to verify their email address.
12. The user verifies their email address.
13. The application confirms the email verification.
14. The application creates the user account in the database.
15. If any account does not exist for the email address entered by user, account is created successfully, and the user is redirected to Dashboard screen.
16. If an account already exists with the email address entered by the user, the user is informed that an account already exists and is redirected to the registration screen.

6.1.2. Login

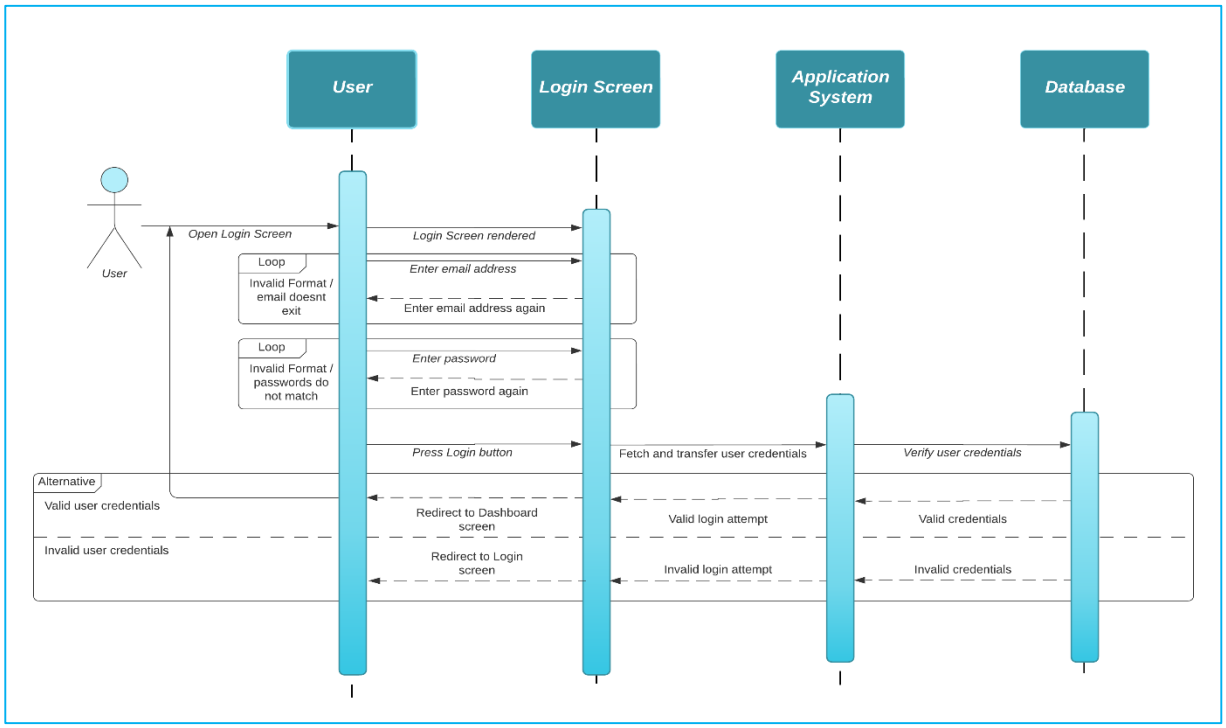

Figure 11: Login Sequence Diagram

1. User opens the Login screen.
2. Login screen is displayed.
3. User enters their email address.
4. If the format for email address is invalid, user enters email address again.
5. User enters their password.
6. If the format for password is invalid, user enters password again.
7. User presses login button.
8. User data is fetched from Login screen and transferred to the application system.
9. The application system verifies user credentials from database.
10. If the user credentials are valid then the user is redirected to their dashboard screen.
11. If the user credentials are not valid then the user is redirected to the login screen.

6.1.3. Logout

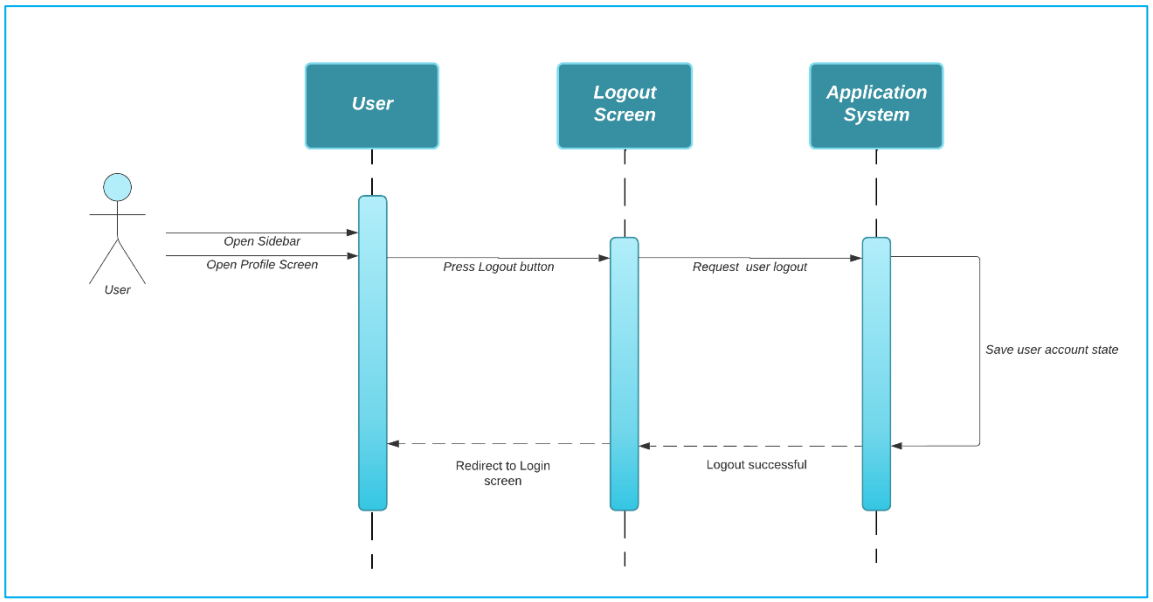

Figure 12: Logout Sequence Diagram

1. User may open their profile screen or the sidebar.
2. User presses the logout button.
3. The logout screen sends a request to the application system for user logout.
4. The application system saves the state of the user account.
5. The user is redirected to the Login screen.

6.1.4. Become a tutor

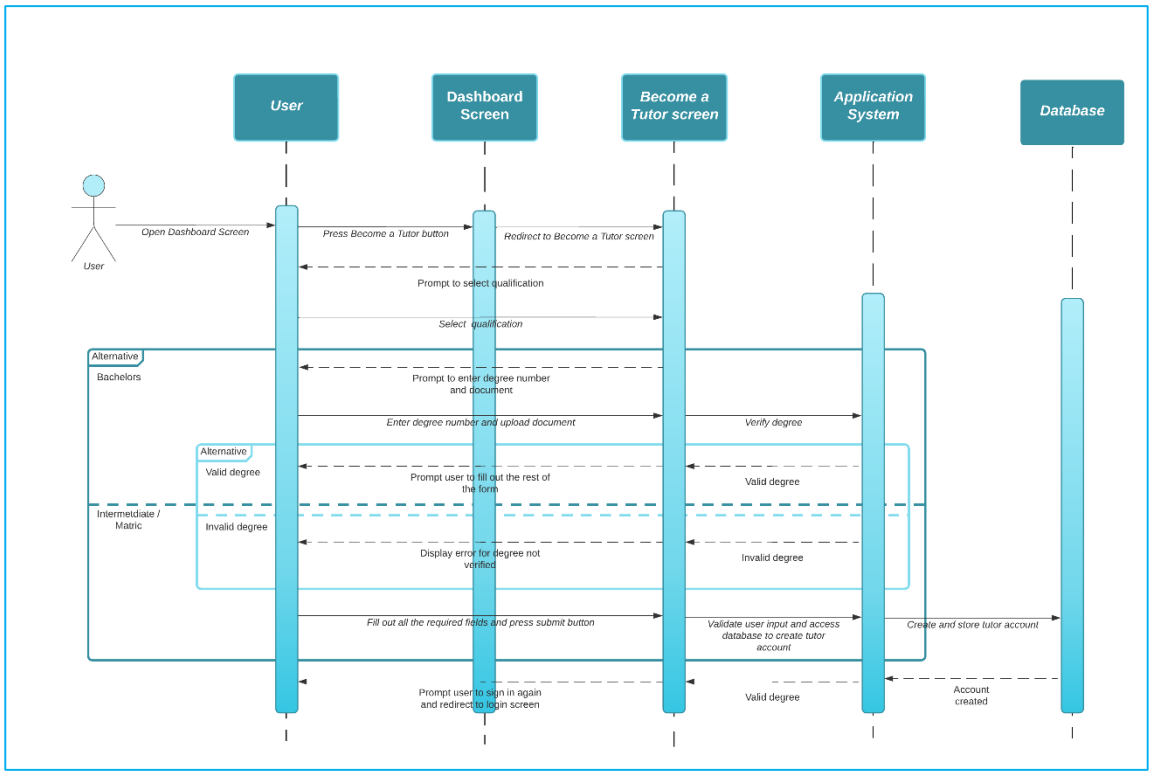

Figure 13: Become a Tutor Sequence Diagram

1. The student opens their dashboard screen.
2. The student presses the Become a Tutor button.
3. The user is redirected to the Become a Tutor screen.
4. The user is prompted to select a qualification level.
5. If the user selects “Bachelors” as qualification level, they will be prompted to enter their degree details:
    a. The user enters their degree details.
    b. The application verifies the user’s degree.
    c. If the user’s degree is verified successfully, the user then fills the rest of the fields of the form.
    d. If the user’s degree is *not* verified, the application displays error for invalid degree to the user.
6. If the user selects “Intermediate” or “Matric” as qualification level, the degree verification stage will be omitted.
7. The application validates the input and accesses the database to create user account.
8. After successful account creation, the user is prompted to re-login to the application as a tutor.

6.2. State Diagram

6.2.1. Meeting

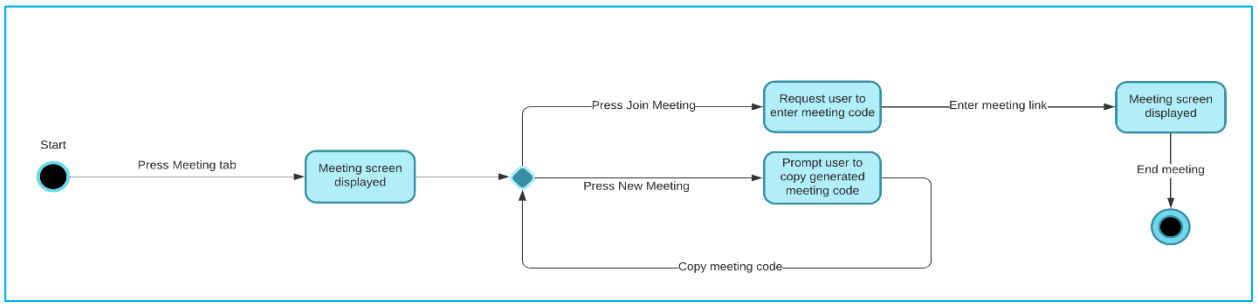

Figure 14: Meeting State Diagram

1. The user presses the meeting tab from the bottom navigation bar.
2. The meeting screen is displayed.
3. If the user presses New Meeting button:
    a. The user is prompted to copy the generated meeting code.
    b. The user copies the code.
    c. The user then presses the Join Meeting button.
    d. The user pastes the code and joins the meeting.
    e. Meeting screen is displayed.
4. If the user presses the Join Meeting button:
    a. The user pastes the code provided to them previously by the tutor.
    b. The user then joins the meeting.
    c. Meeting screen is displayed.

6.2.2. Search for tutors

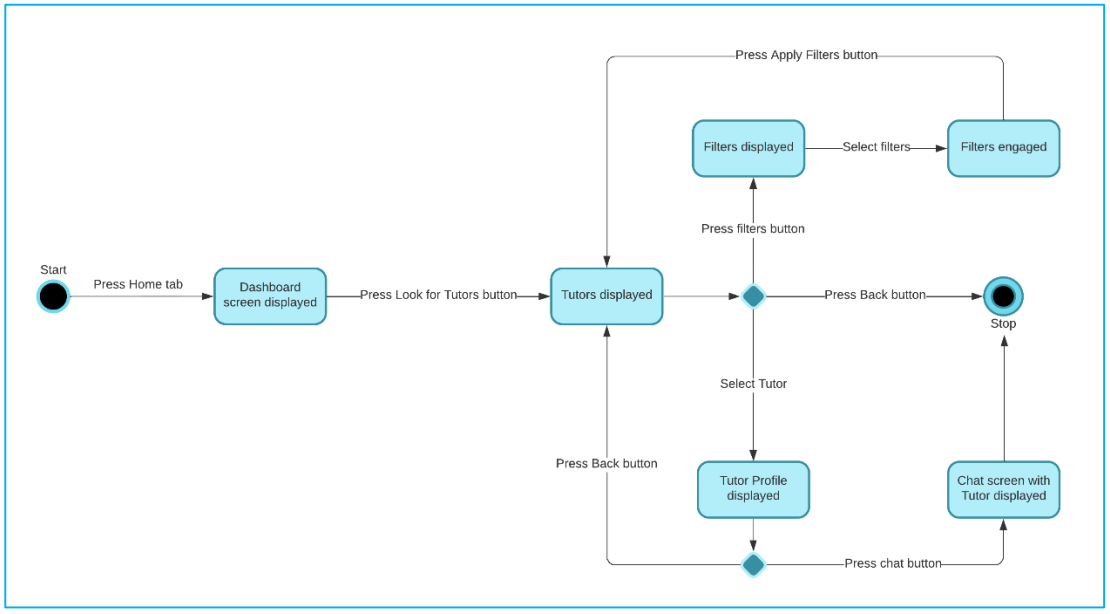

Figure 15 Search for Tutors State Diagram

1. The user presses the Search for Tutors button from the Dashboard screen.
2. The tutor list is displayed.
   a. The user presses filters button:
      i. The filter screen is displayed.
      ii. The users enter or selects their desired filters.
      iii. The user presses Apply Filters button.
      iv. The filtered tutors are displayed.
   b. The user selects a Tutor:
      i. Tutor profile is displayed.
         1. The user presses back button.
         2. The user is directed back to [2].
      ii. The user presses Message button.
         1. Chat screen with Tutor is displayed.

6.2.3. Edit Profile

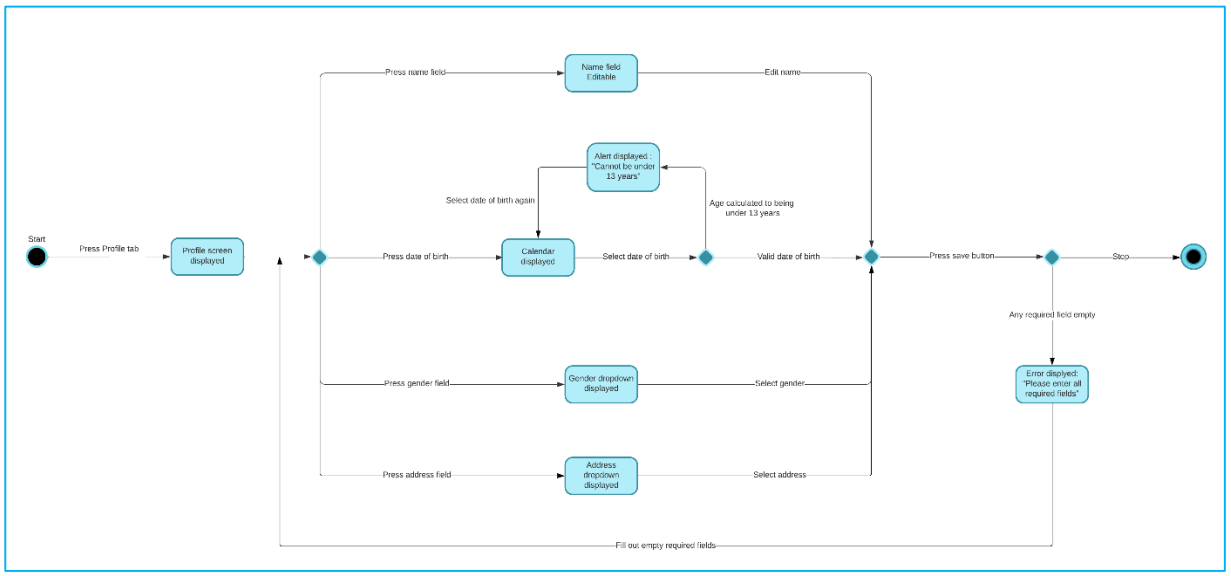

Figure 16: Edit Tutor State Diagram

1. The user presses the Profile icon on the Dashboard.
2. Profile screen is displayed.
   a. Edit Name
      i. Name field becomes editable.
   b. Edit Date of Birth
      i. The calendar is displayed.
         1. The user selects the date of birth.
         2. If the age is below 13 years, the user is prompted to re-enter the DOB.
   c. Edit Gender
      i. User can select the gender from the dropdown menu.
   d. Edit Address

i. User can select the address from the dropdown menu.

3. User presses Save button.
    a. If any required field is empty, the user is promoted to enter all the required fields.

7. Testing and Evaluation

7.1. Test cases

7.1.1. Sign up

TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
SU01	Check customer sign up with valid data type	1. Go to sign up form 2. Enter relevant fields 3. Click Sign up	Name: Hassan Jamil Date of Birth: 14-07-2001 Password: testabcd123 Confirm Password: testabcd123 Location: Block 15, Gulistan e Johar	User should go smoothly with all steps and sign up successfully.	Same as expected	Pass
SU02	Check customer sign up with invalid data type	1. Go to sign up form 2. Enter relevant fields 3. Click Sign up	Name: H@ssan Jamil Date of Birth: 14-07-2001 Password: testabcd123 Confirm Password: testabcd123 Location: Block 15, Gulistan e Johar	User should be warned of the incorrect data format and not proceed to successful Sign up.	Same as expected	Pass
SU03	Check customer sign up with valid password format	1. Go to sign up form 2. Enter password 3. Fill other fields 4. Click Sign up	Password: jaMil420	User should go smoothly with all steps and sign up successfully.	As expected	Pass
SU04	Check customer sign up with	1. Go to sign up form	Password: @1234	User should be warned, as password	As expected	Pass

	invalid password format	2. Enter password 3. Fill other fields Click Sign up		criteria allows minimum 8 characters, with at least 1 upper and lower case letter and number.		

7.1.2. Login

TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
LG01	Check User Login with valid Data	1. Go to Log in form 2. Enter User Email 3. Enter User Password 4. Click Log in	Email: testemail@gmail.com User password: aBcd1234	Login should be successful.	As expected	Pass
LG02	Check User Login with invalid Data	1. Go to Log in form 2. Enter User Email 3. Enter User Password 4. Click Log in	Email: testemail@gmail.com User password: pqr123	Login should not be successful as password is incorrect for the entered email	As expected	Pass
LG03	Check User Login if one of the fields is left empty	1. Go to Log in form 2. Skip either user email or password Click Log in	User email: testemail@gmail.com User password: ‖ User ID: User password: aBcd1234	User should be prompted with the error and instructed to re-enter both values.	As expected	Pass

7.1.3. Become a tutor

TEST CASE ID	DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS /FAIL
BT01	Check if user can sign up as a tutor while being a student	1. Open the become a tutor form 2. Enter tutoring details, including their schedule and available days, and their degree document.	Based on relevant fields selected, qualification, degree document, subjects, rates, days available.	No error prompted.	As expected	Pass
BT02	Check if user can sign up to be a tutor if they are a tutor	1. Attempt to navigate to become a tutor section	None	Screen not displayed, as it is not possible to reapply	As expected	Pass
BT03	Check if data entered is verified before entry	1. Navigate to become tutor screen 2. Attempt to fill invalid details	Relevant to fields selected, example: Not selecting a day for availability	User should be prompted with error that forces correction	As expected	Pass

7.1.4. Send Message

TEST CASE ID	DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
SM01	Check if message sending is received at both sends	1. User views list of tutors, selects one and clicks the contact them option 2. User sends them a	Text: “Hey there”	This message should be visible to the user	As expected	Pass

		message				
SM02	Check if message sending is viewed appropriately, not warped out of screen	1. Users send and receive messages	Text: “Hey There” “Hello” “How are you?” “Well, what about you?”	Text should appear colored individually depending on sender and receiver at both ends, and not overflow the screen	As expected	Pass

7.1.5. View/Edit User Info

TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
VEU01	User updates their user information	1. Go to profile 2. Modify relevant fields 3. Click save	Name: Sarim CHANGED TO Name: Sameer	User should go smoothly with all steps and update info successfully.	As expected	Pass
VEU02	User updates their user information with invalid data format	1. Go to profile 2. Modify relevant user information 3. Click save	Name: Sarim CHANGED TO Name: S@meer	User should be shown an error, that tells them this format is wrong.	As expected	Pass

7.1.6. Video Call

TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
VC01	Check if user is able to create a meeting link and join it	1. User goes to video call via navigation 2. User generates a link 3. User copies and pastes the link 4. User joins the meeting	Room Code: abc-def-ghy	The user should successfully join the meeting	As expected	Pass
VC02	Check if user is able to join an invalid meeting code	1. User goes to video call via navigation 2. User generates a link 3. User copies and pastes the link, but adds a typo instead 4. User joins the meeting	Room Code: abc-def-ghzy	The meeting should not start as no meeting link like that exists	As expected	Pass
VC03	Check if multiple users can join a video call	1. User goes to video call via navigation 2. User generates a link 3. User copies and pastes the link, and sends it to others via the messaging feature beforehand 4. Users joins the meeting	Room Code: abc-def-ghy	The meeting should start and multiple users should appear and be able to talk to teach other	As expected	Pass

7.1.7. Send Contract

TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
SC01	Check if user is able to send a contract to a tutor	1. User views current tutor list, selects one 2. User clicks the send contract button 3. User can then send a contract to the tutor after filling out relevant details	Subjects: Maths, Chemistry Days: Monday, Tuesday, Friday Timings: 9-10 AM, 11-12PM	The contract should be successfully sent to the tutor, and will show varying data when the tutor is different	As expected	Pass
SC02	Check if user can send invalid data or manipulate data in contract	1. User views current tutor list, selects one 2. User clicks the send contract button 3. User can then send a contract to the tutor after filling out relevant details	User selects ,deselects time slots, subjects, days, to mess with the calculated price	The price should account for all form changes instantaneously and disallow manipulation	As expected	Pass

7.1.8. View Students/Contract

TEST CASE ID	TEST CASE DESCRIPTION	TEST STEPS	TEST DATA	EXPECTED RESULTS	ACTUAL RESULT	PASS/FAIL
VSC01	Check if user can view other contracts apart from his own, or students apart from his own	1. User, who is a tutor, can view their current contracts and session dates	NA	The user should only be able to see their own data since the database query only works for specific users	As expected	Pass
VSC02	Check if user can view other student data or information	1. User navigates to contract 2. User clicks view students button	NA	The user should only be able to see the data of students who have contracted them	As expected	Pass

7.2. Results

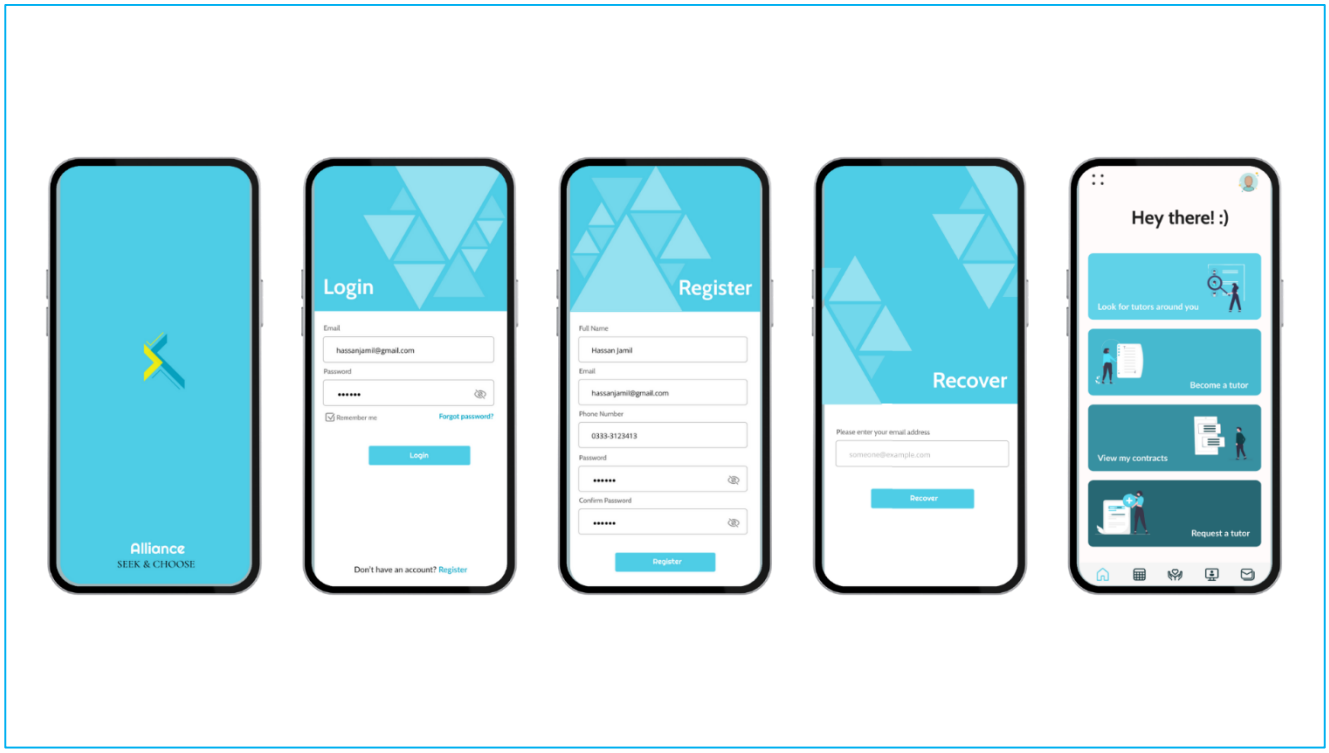

Figure 17: Welcome, Login/Signup, Recover, Dashboard Screens

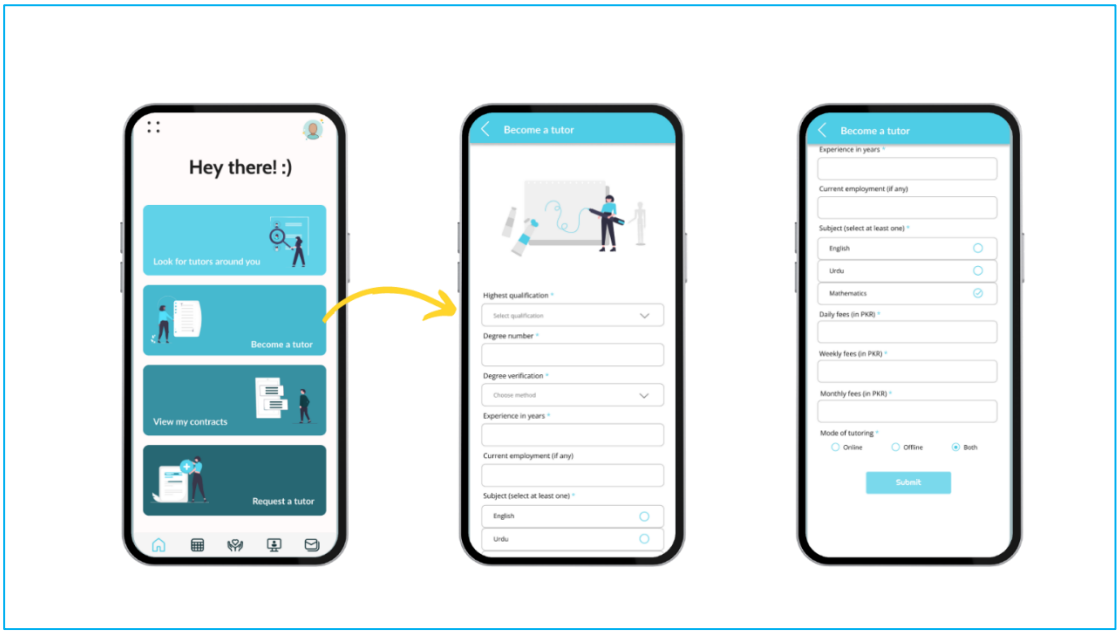

Figure 18: Become a tutor Screen

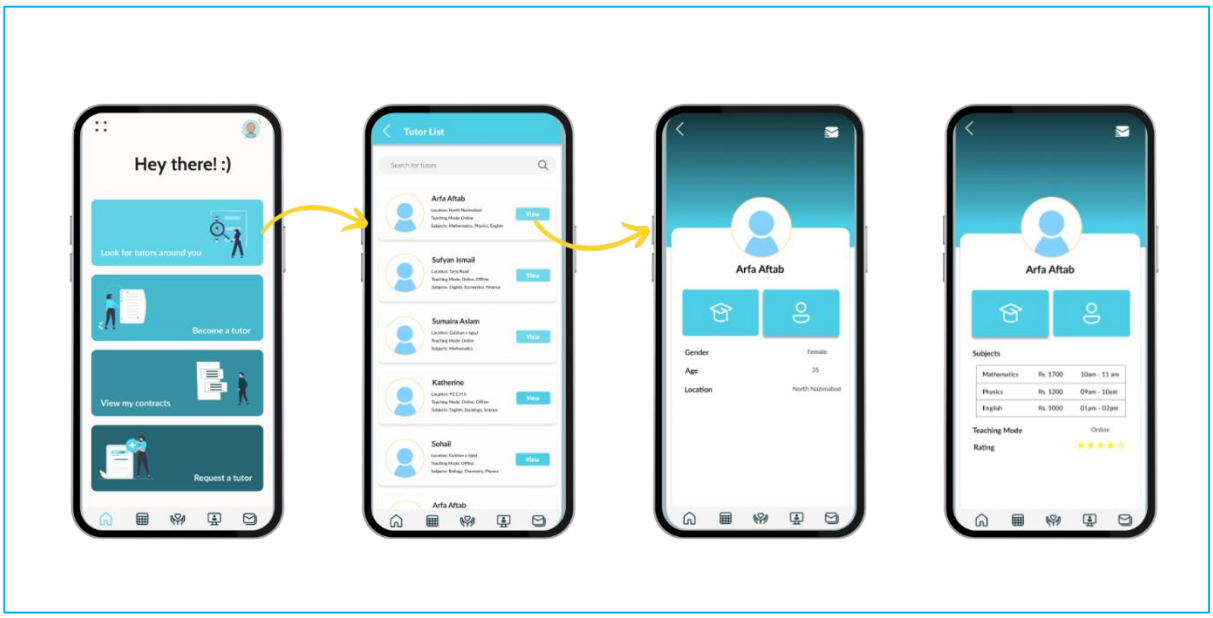

Figure 19: Search for tutors, Tutor List, Tutor Profile Screens

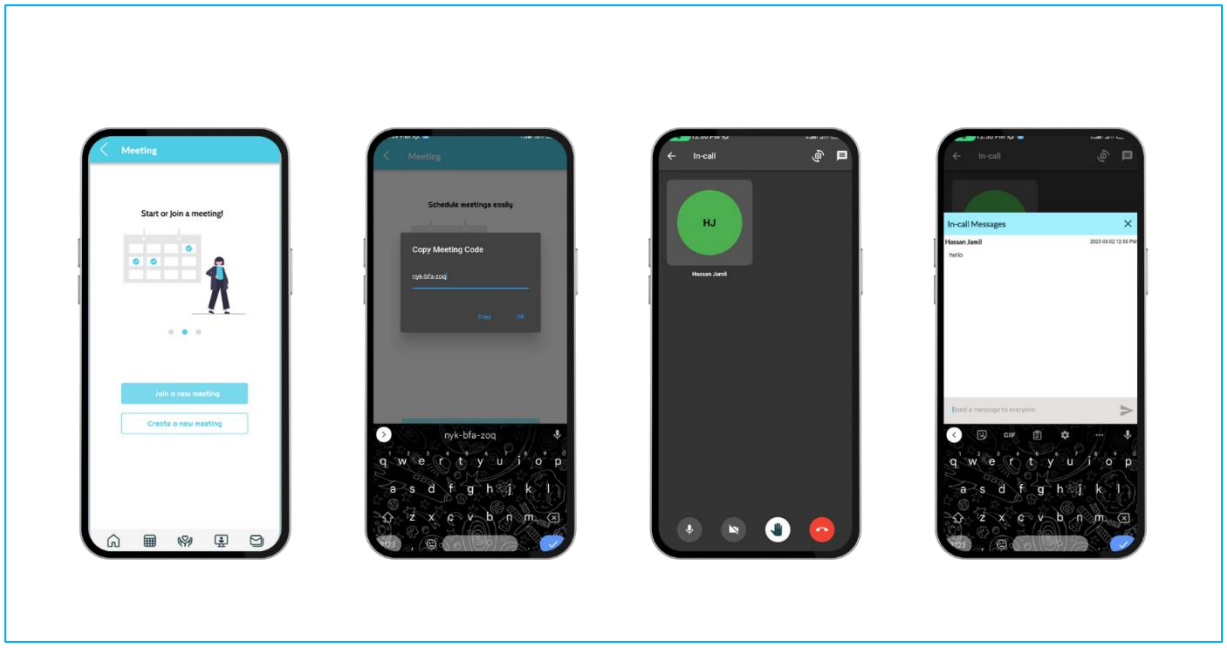

Figure 20: Join a meeting, Create a new meeting Screens

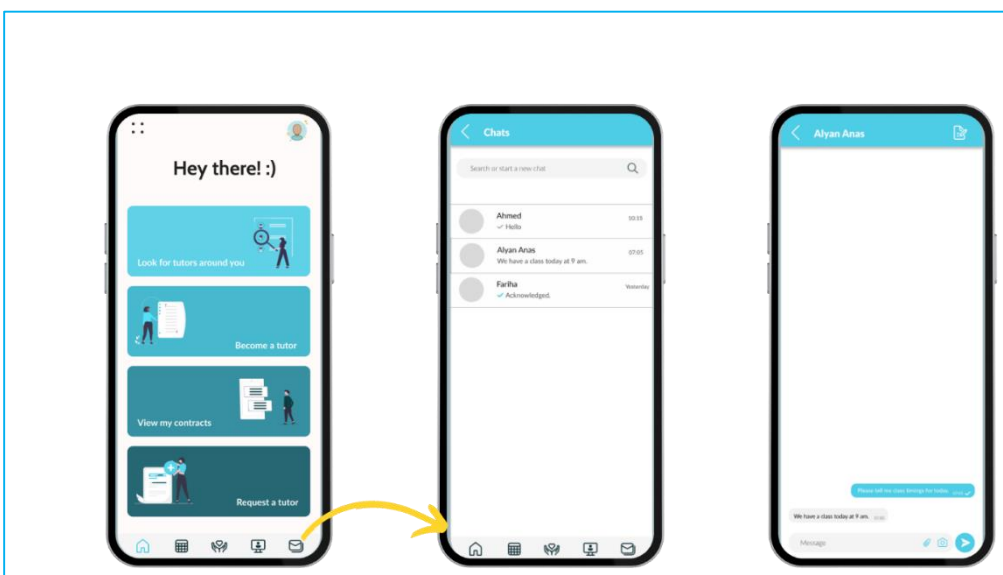

Figure 21: Chat Screen

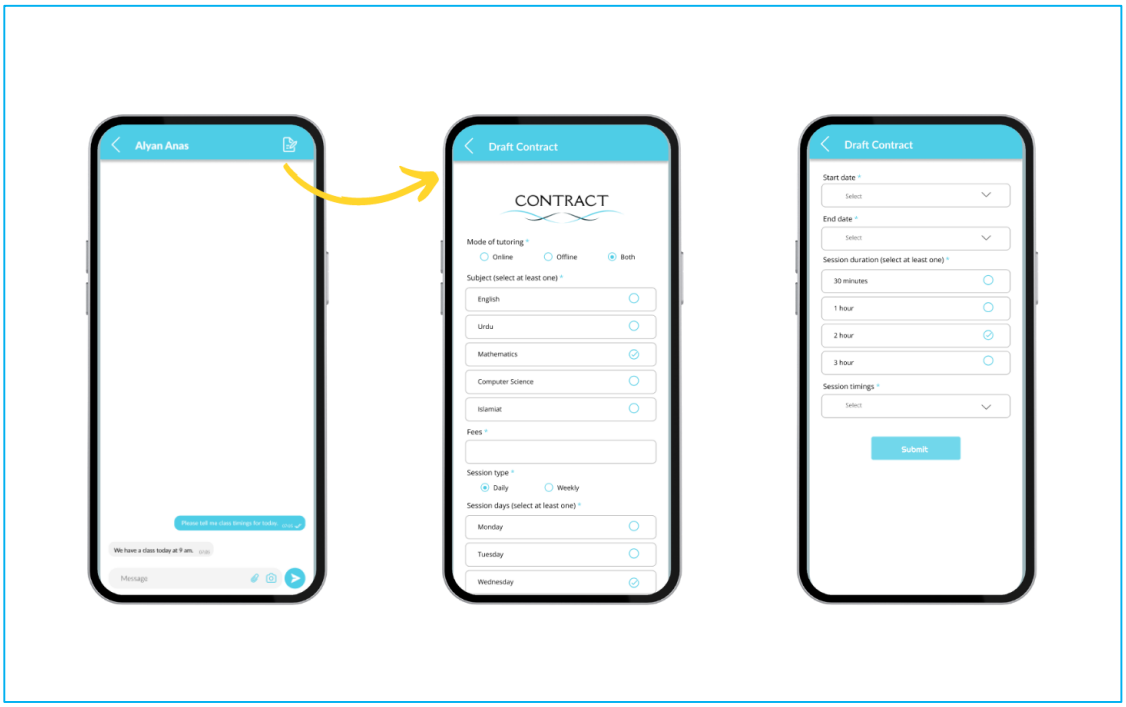

Figure 22 Draft a contract Screen

8. Conclusion

In conclusion, Alliance is a comprehensive and innovative mobile application that connects students in Karachi with qualified tutors for academic support. Built using the Flutter framework and Firebase backend services, the app provides an intuitive user interface and seamless functionality, including in-app messaging, video conferencing, and a contract system to facilitate tutor-student interactions. With a focus on personalized learning and tutor recruitment and onboarding, Alliance aims to increase educational volunteer work and provide a safe and reliable means of tutoring to students in need. By addressing the challenges of accessing quality education, particularly in the context of remote learning, Alliance represents a promising solution for the students and tutors of Karachi.

9. References

after_layout: https://pub.dev/packages/after_layout

cached_network_image: https://pub.dev/packages/cached_network_image

carousel_slider: https://pub.dev/packages/carousel_slider

cloud_firestore: https://pub.dev/packages/cloud_firestore

country_state_city_picker: https://pub.dev/packages/country_state_city_picker

cupertino_icons: https://pub.dev/packages/cupertino_icons

dropdown_button2: https://pub.dev/packages/dropdown_button2

dropdown_search: https://pub.dev/packages/dropdown_search

firebase_auth: https://pub.dev/packages/firebase_auth

firebase_core: https://pub.dev/packages/firebase_core

firebase_storage: https://pub.dev/packages/firebase_storage

flutter_calendar_carousel: https://pub.dev/packages/flutter_calendar_carousel

flutter_date_range_picker: https://github.com/linagora/flutter-date-range-picker

flutter_mobx: https://pub.dev/packages/flutter_mobx

flutter_rating_bar: https://pub.dev/packages/flutter_rating_bar

flutter_secure_storage: https://pub.dev/packages/flutter_secure_storage

flutterfire_cli: https://pub.dev/packages/flutterfire_cli

google_sign_in: https://pub.dev/packages/google_sign_in

group_radio_button: https://pub.dev/packages/group_radio_button

hmssdk_flutter: https://pub.dev/packages/hmssdk_flutter

html: https://pub.dev/packages/html

http: https://pub.dev/packages/http

image_cropper: https://pub.dev/packages/image_cropper

image_picker: https://pub.dev/packages/image_picker

intl: https://pub.dev/packages/intl

mobx: https://pub.dev/packages/mobx

mobx_codegen: https://pub.dev/packages/mobx_codegen

permission_handler: https://pub.dev/packages/permission_handler

responsive_framework: https://pub.dev/packages/responsive_framework

shared_preferences: https://pub.dev/packages/shared_preferences

time_machine: https://pub.dev/packages/time_machine

uuid: https://pub.dev/packages/uuid

xpath: https://pub.dev/packages/xpath

shimmer: https://pub.dev/packages/shimmer