

GRAD CHAT

A PROJECT REPORT

Submitted by

PRANITHA P
(Reg. No: 24MCR076)

SAKTHIVEL R
(Reg. No: 24MCR086)

SAM ANDERSON Y
(Reg. No: 24MCR087)

*in partial fulfilment of the requirements
for the award of the degree*

of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS



KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER 2024

DEPARTMENT OF COMPUTER APPLICATIONS**KONGU ENGINEERING COLLEGE****(Autonomous)****PERUNDURAI, ERODE – 638 060****DECEMBER 2024****BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**GRAD CHAT**” is the bonafide record of project work done by **PRANITHA P (Reg. No: 24MCR076)**, **SAKTHIVEL R (Reg. No: 24MCR086)**, **SAM ANDERSON Y (Reg. No: 24MCR087)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

SUPERVISOR**HEAD OF THE DEPARTMENT****(Signature with seal)****Date:**

Submitted for the end Semester viva voce examination held on _____

INTERNAL EXAMINER**EXTERNAL EXAMINER**

DECLARATION

We affirm that the project report entitled “**GRAD CHAT**” being submitted in partial fulfilment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidates.

Date

PRANITHA P

(Reg. No: 24MCR076)

SAKTHIVEL R

(Reg. No: 24MCR086)

SAM ANDERSON Y

(Reg. No: 24MCR087)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date:

Name and Signature of the supervisor

(Ms.T.KALPANA)

ABSTRACT

The Junior-Senior Mentorship App for Navigating College Success *Together* is an innovative mobile application designed to bridge the gap between college juniors and seniors, fostering a culture of mentorship and mutual growth. This platform empowers juniors to seek guidance and insights while enabling seniors to share their knowledge, creating a collaborative and supportive college environment.

The app features two distinct user profiles tailored to the needs of juniors and seniors. The Junior Profile focuses on assisting new students, especially introverts, in finding guidance on critical topics like placement strategies, effective study techniques, event management, and academic excellence. Meanwhile, the Senior Profile provides a space for experienced seniors and super seniors to mentor juniors by sharing valuable resources, announcing placement opportunities, and organizing skill development events.

With a user-friendly interface, the application allows juniors to connect with seniors based on specific categories, including placement mentors, event coordinators, academic achievers, and super seniors who have secured prestigious placements. This categorization ensures juniors can find the right mentors easily and efficiently.

Developed using React Native for the front end, JavaScript as the coding language, and Firebase for the back end, the app leverages modern technology to deliver seamless performance and accessibility. By promoting meaningful interactions and knowledge sharing, this app not only simplifies mentorship but also strengthens the college community, ensuring students navigate their academic journey with confidence and success.

ACKNOWLEDGEMENT

We respect and thank our correspondent **Thiru.A.K.ILANGO BCom., MBA., LLB.,** and our Principal **Dr.V.BALUSAMY BE(Hons)., MTech., PhD** Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of the Department **Dr.A.TAMILARASI Msc., MPhil., PhD., MTech.,** Department of Computer Applications, Kongu Engineering College for her perfect guidance and support that made this work to be completed successfully.

We also like to express our gratitude and sincere thanks to our project coordinators **Mrs.S.HEMALATHA MCA.,** Assistant Professors(Sr.G), Department of Computer Applications, Kongu Engineering college who have motivated us in all aspects for completing the project in scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Mrs.T.KALPANA MCA.,** Assistant Professor, Department of Computer Applications, Kongu Engineering College for giving his valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping us to overwhelm in all proceedings. We bow our heart and head with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	ACKNOWLEDGEMENT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 ABOUT THE PROJECT	1
	1.2 EXISTING SYSTEM	2
	1.3 DRAWBACKS OF EXISTING SYSTEM	2
	1.3 PROPOSED SYSTEM	3
	1.4 ADVANTAGES OF THE PROPOSED SYSTEM	3
2	SYSTEM ANALYSIS	4
	2.1 IDENTIFICATION OF NEED	4
	2.2 FEASIBILITY STUDY	4
	2.2.1 Technical Feasibility	5
	2.2.2 Operational Feasibility	5
	2.2.3 Economical Feasibility	5
	2.3 SOFTWARE REQUIREMENT SPECIFICATION	6
	2.3.1 Hardware Requirements	6
	2.3.2 Software Requirements	7
3	SYSTEM DESIGN	12
	3.1 MODULE DESCRIPTION	12
	3.2 DATA FLOW DIAGRAM	14
	3.3 DATABASE DESIGN	18

	3.4 INPUT DESIGN	20
	3.5 OUTPUT DESIGN	24
4	IMPLEMENTATION	26
	4.1 CODE DESCRIPTION	26
	4.2 STANDARDIZATION OF THE CODING	26
	4.3 ERROR HANDLING	26
5	TESTING AND RESULTS	29
	5.1 TESTING	29
	5.1.1 Unit Testing	29
	5.1.2 Integration Testing	31
	5.1.3 Validation Testing	32
6	CONCLUSION AND FUTURE ENHANCEMENT	33
	6.1 CONCLUSION	33
	6.2 FUTURE ENHANCEMENT	33
	APPENDICES	35
	A. SAMPLE CODING	35
	B. SCREENSHOTS	45
	REFERENCES	54

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Dataflow Diagram Level 0	14
3.2	Dataflow Diagram Level 1	15
3.3	Dataflow Diagram Level 2	16
3.4	Use Case Diagram	17
3.5	Class diagram	18
3.4	Login Screen	22
3.5	Senior Register Screen	23
3.6	Junior Register Screen	24
3.7	Storing in Firebase	25
3.8	Community Chat Stored in Firebase	26
B.1	Senioe & Junior Register Screen	45
B.2	Login Screen	46
B.3	Community Chat Screen	47
B.4	Home Screen	48
B.5	AI Chatbot	49
B.6	Placement User Screen	50
B.7	Profile Screen	51
B.8	Profile Management Screen	52
B.9	Firestore database	53

LIST OF ABBREVIATIONS

DB	Data Base
GB	Gigabyte
DFD	Data Flow Diagram
SDK	Software Development Kit
DOM	Document Object Model
NPM	Node Package Manager
API	Application Programming Interface
XML	Extensible Markup Language
AI	Artificial Inteligence
MFA	Multi-Factor Authentication
RTDB	Real-Time Database
JSON	JavaScript Object Notation
IDE	Integrated DevelopmentEnvironment
UX	User Experience

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

The purpose of this project is to develop Grad-Chat, a mobile application designed to facilitate effective mentorship and collaboration between college juniors and seniors. Traditional mentorship systems in educational institutions often rely on informal communication channels, leading to inefficiencies, missed opportunities, and lack of structure. Grad-Chat aims to address these issues by providing a unified digital platform to streamline mentorship.

The application enables juniors to seek guidance on academics, placements, and event management, while allowing seniors to share resources, post updates, and organize activities. Features like real-time communication, categorized mentor profiles, and progress tracking ensure personalized and efficient mentorship experiences. The platform also integrates notification systems and user-friendly navigation to keep users informed and engaged.

Grad-Chat promotes accessibility by bridging the gap between juniors and seniors, fostering a supportive ecosystem for sharing knowledge and building connections. By offering a structured approach to mentorship, the platform empowers users to maximize their academic and professional potential while building a sense of community.

Overall, Grad-Chat simplifies mentorship processes, enhances transparency, and encourages collaboration, making it an invaluable tool for educational institutions. It supports users in achieving personal growth, skill development, and meaningful interactions, creating a long-lasting impact on student success and engagement.

1.1 EXISTING SYSTEM

In the existing system for college mentorship, there is no dedicated platform for facilitating communication between juniors and seniors. Mentorship often relies on word of mouth, informal connections, and manual coordination, which can be inefficient and unorganized. Juniors struggle to find the right mentors, while seniors find it difficult to share their knowledge. The process is time-consuming and lacks the structure needed to manage mentorship opportunities, making it harder to track progress and provide consistent guidance.

1.2 DRAWBACK OF EXISTING SYSTEM

The existing mentorship system faces several challenges due to its informal and unstructured nature, which leads to inefficiencies and difficulties in management. These drawbacks include:

- Lack of a centralized platform for mentorship, causing difficulties in connecting juniors with the right seniors.
- Time-consuming manual coordination for mentorship requests and responses.
- Inefficient tracking of progress and communication, often resulting in missed opportunities.
- Difficult to maintain and update mentorship data, such as mentee-mentor interactions.
- Limited accessibility to resources and opportunities for juniors, leading to unorganized guidance.
- Absence of real-time updates, resulting in delayed responses and guidance.
- Lack of categorization for mentor profiles, making it hard for juniors to find the right support.

1.3 PROPOSED SYSTEM

The proposed Grad-Chat system simplifies mentorship by providing a centralized platform for seamless communication between juniors and seniors. It features categorized mentor profiles, real-time updates, notifications, and secure data handling. The user-friendly interface enables easy navigation, while progress tracking and feedback ensure

continuous improvement. Grad-Chat aggregates resources, events, and opportunities, making mentorship efficient and accessible, empowering users to achieve academic and professional growth conveniently via mobile devices.

1.4 ADVANTAGES OF THE PROPOSED SYSTEM

- The system provides a user-friendly interface, making it easy for both juniors and seniors to navigate and use the platform effectively.
- It streamlines the mentorship process, making it easier and more flexible for juniors to connect with the right mentors based on their needs.
- The platform can be accessed anytime and anywhere via mobile devices, ensuring continuous availability of mentorship opportunities.
- Juniors can efficiently find suitable mentors, reducing time spent searching for guidance.
- The system offers a wide range of mentorship categories, allowing juniors to access expertise on academics, placements, event management, and more.

SUMMARY

This chapter provides an overview of the existing mentorship system, its drawbacks, and the proposed system with its advantages. The purpose of this project, *Grad-Chat*, was to develop a mobile application that bridges the gap between college juniors and seniors, fostering mentorship and community engagement. This project allowed us to gain valuable insights and hands-on experience in various areas, including mobile app development using React Native, backend management using Firebase, and the design of user-friendly interfaces for efficient mentor-mentee interactions. The system's functionality, along with the required software and hardware specifications, has been detailed in Chapter 2.

CHAPTER 2

SYSTEM ANALYSIS

2.1 IDENTIFICATION OF NEED

The current mentorship system is inefficient and disorganized, making it difficult for juniors to find suitable mentors and track progress. Manual coordination, separate records, and inefficient communication often result in errors and missed opportunities. A clear need exists for an automated mentorship platform. Grad-Chat addresses this by offering real-time communication, easy mentor search, progress tracking, feedback collection, and automatic notifications. With categorized mentor profiles and a user-friendly interface, Grad-Chat ensures smooth, efficient interactions, making mentorship more accessible, organized, and effective for juniors and seniors.

2.2 FEASIBILITY STUDY

A feasibility study is a complete analysis that is conducted to evaluate the practicality and viability of a proposed project. The prime goal of a feasibility study is to determine whether the project is worth pursuing and if it can be successfully implemented within the defined constraints. Each structure has to be thought of in the developing of the project, as it has to serve the end user in a friendly manner. Three considerations involved in feasibility are

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

2.2.1 TECHNICAL FEASIBILITY

The Grad-Chat mobile application is designed to be accessible on smartphones, allowing users to connect with mentors anytime, anywhere. It doesn't require specialized technology or infrastructure, making it a viable solution for institutions with limited resources. The project leverages widely available tools such as React Native for app development and Firebase for realtime data storage, ensuring the use of widely understood technologies. The technical feasibility also includes ensuring that the system's design is scalable and user-friendly, with minimal technical skill required for access. This lowers barriers for both juniors and seniors who may have varying levels of technical expertise. The technical solution is appropriate for the goals of the system, making it feasible in today's fast-paced educational environment.

2.2.2 OPERATIONAL FEASIBILITY

Operational feasibility evaluates how well the Grad-Chat system will support both users and administrators. The proposed system aims to resolve the current mentorship issues by streamlining communication and simplifying mentor-mentee matching. With a user-friendly interface, even those with basic technical knowledge can easily navigate the platform, ensuring accessibility for a wide range of users. Grad-Chat is designed to integrate seamlessly with existing organizational structures and processes, improving efficiency and reducing operational complexities. It will enhance mentorship effectiveness by simplifying the tracking of interactions, feedback, and resource sharing. Overall, the system is expected to have a positive impact on user satisfaction, making mentorship more accessible and organized within the institution.

2.2.3 ECONOMIC FEASIBILITY

The economic feasibility of the Grad-Chat project considers costs for mobile app development, backend infrastructure (Firebase), and cloud services. Transitioning from a manual mentorship system to a digital platform will significantly reduce administrative overhead, as it automates the matching process, communication, and progress tracking. This will result in a reduction of manual errors and time-consuming coordination tasks, leading to long-term operational cost savings. Additionally, the system will improve

mentorship quality by fostering better connections between mentors and mentees, which in turn enhances student engagement and satisfaction. By streamlining communication and providing a centralized platform for all mentorship-related activities, Grad-Chat offers a cost-effective solution that can reduce the need for physical meetings and resources. The resulting improvements in mentorship efficiency and engagement will provide increased value to both students and the institution, ensuring substantial return on investment.

2.3 SOFTWARE REQUIREMENT SPECIFICATION

A system requirement defines the capabilities a system must possess to meet user needs. It includes hardware, software, and business operation specifications, detailing what is necessary for the system to function effectively. These requirements guide the development process, ensuring the system operates as intended while optimizing costs. By addressing user needs efficiently, system requirements help in reducing implementation complexity and improving overall system performance.

2.3.1 HARDWARE REQUIREMENTS

The hardware for the system is selected considering the factors such as CPU processing speed, memory access, peripheral channel access speed, printed speed; seek time& relational dataof hard disk and communication speed etc.

Processor	:	Intel Core i5 or higher
RAM	:	8GB DDR4 or higher
Monitor	:	15” Full HD Monitor
Hard Disk	:	512GB SSD or higher
Keyboard	:	Standard keyboard (preferably with backlight)

2.3.2 SOFTWARE REQUIREMENTS

The software for the project is selected considering the factors such as working frontend environment, flexibility in the coding language, database knowledge of enhances in backend technology etc.

Operating System	:	Windows, linux, mac
Language	:	React Native, JavaScript
Backend	:	Firebase (for real-time data handling)
Database	:	Firebase (for data storage and real-time updates)

FRONT END

React Native is an open-source framework developed by Facebook that enables the creation of mobile applications using JavaScript and React. It allows developers to build natively-rendered mobile applications for iOS and Android platforms from a single codebase, which reduces development time and resources. React Native provides a seamless user experience by utilizing native components, ensuring high performance and responsiveness across platforms.

The core of React Native's architecture is its use of JavaScript and React to build mobile UIs. React Native utilizes a virtual DOM and components, which simplifies the development process and enhances the maintainability of code. Components in React Native are reusable and can be combined to form complex UIs, making it a modular and efficient framework for building mobile apps.

React Native also leverages native modules, allowing developers to access platform-specific functionality like GPS, camera, and sensors while writing most of the code in JavaScript. This enables a high level of customization while maintaining a consistent user experience. Moreover, React Native supports hot reloading, which enables developers to instantly see code changes in real-time without losing the app's state, thus speeding up the development cycle.

The framework integrates well with popular backend services like Firebase, enabling easy management of data, authentication, and real-time updates. React Native also offers a rich ecosystem of libraries and tools, providing ready-to-use solutions for common development tasks such as navigation, state management, and data handling. This flexibility and community support make React Native a powerful tool for building high-quality, cross-platform mobile applications.

Features of React Native

- **Cross-Platform Development:** React Native enables developers to build mobile applications for both iOS and Android using a single codebase, which reduces development time and costs.
- **Hot Reloading:** This feature allows developers to instantly see changes in the code without rebuilding the entire application, speeding up the development process.
- **Native Components:** React Native provides native components like buttons, sliders, and text inputs, ensuring that apps have a native look and feel with smooth, responsive user interfaces on both platforms.
- **Declarative UI:** Developers can define UI components declaratively. This results in easier maintenance and debugging as the UI automatically updates when data changes, and the code can be broken into smaller, reusable components.
- **Third-Party Plugin Support:** React Native supports the use of third-party plugins, allowing developers to extend app functionality by integrating features like geolocation or payment gateways.
- **Performance Optimization:** React Native uses a virtual DOM to update only the components that have changed, improving the app's performance by ensuring faster and more efficient rendering.

- **Access to Native APIs:** React Native provides direct access to device APIs and native code, allowing developers to use platform-specific features and optimize app performance where necessary while maintaining a native feel.

Back End

Firebase is a comprehensive, open-source (partially), and cross-platform Backend-as-a-Service (BaaS) platform developed by Google. It provides ready-to-use services for building scalable, real-time back-end solutions without managing servers. Firebase is not a programming language or framework but a suite of cloud-based tools that handle data storage, authentication, hosting, and more.

It is widely used for building back-end services for web, mobile, and serverless applications. Large organizations, including Google, Alibaba, Lyft, and The New York Times, rely on Firebase for its performance and scalability.

Features of Firebase

- **Real-Time Data Synchronization:** Firebase uses Realtime Database and Cloud Firestore to sync data in real time across all connected clients, enabling fast and interactive applications
- **Event-Driven:** Firebase works asynchronously and supports event-driven architecture with real-time triggers, allowing applications to update data instantly without polling servers.
- **Highly Scalable:** Firebase leverages Google Cloud infrastructure, making it highly scalable and capable of handling millions of concurrent users without manual intervention.
- **No buffering:** Firebase handles data efficiently by enabling instant data transfer and streaming, ideal for applications with audio, video, or live content.
- **Open source tools:** While Firebase itself is proprietary, many Firebase libraries and SDKs are open source and actively maintained by the community.
- **License:** Firebase SDKs and tools are available under open-source licenses.

Firestore

Firestore is a powerful cloud-based NoSQL database system that provides scalable and flexible data management for real-time applications. It includes two primary database options: Firestore Realtime Database and Cloud Firestore, both designed to handle unstructured or semi-structured data efficiently. Firestore stores data in a JSON-like structure, allowing developers to work with flexible schemas that evolve as the application requirements change.

Unlike traditional relational databases, Firestore is schema-less, which makes it easier to adapt to changing data models without significant overhead. Firestore ensures high performance for real-time data synchronization by leveraging a WebSocket-based connection. It supports horizontal scaling through Google Cloud infrastructure, enabling applications to handle heavy loads efficiently by distributing data across servers.

Features of Firestore

- **JSON-Like Data Model:** Firestore stores data in a hierarchical JSON structure (Realtime Database) or as documents grouped into collections (Cloud Firestore). This makes it easy for developers to store and access data intuitively without rigid tables or relationships.
- **Sharding and Horizontal Scaling:** Firestore's databases can scale horizontally by distributing data across multiple servers automatically through Google Cloud infrastructure, ensuring high availability and performance even under heavy loads.
- **Replication:** Firestore ensures data redundancy and fault tolerance through replication. Data is automatically replicated across regions, reducing points of failure and ensuring that applications remain available even during server crashes or hardware issues.
- **Time Series Data:** Firestore can handle time-stamped data effectively, such as logs, metrics, or sensor data. Cloud Firestore's support for range queries and timestamp fields allows efficient management and querying of time-series data.

SUMMARY

he system requirements for the Grad-Chat application are thoroughly detailed, covering the hardware and software components essential to meet both user and system needs. The features of React Native, which plays a significant role in the development, have been discussed. Detailed information regarding the system design will be presented in Chapter 3.

CHAPTER 3

SYSTEM DESIGN

3.1 MODULE DESCRIPTION

A module description provides detailed information about the module and its supported components, which is accessible in different manners.

The project contains the following module:

- Junior & Senior Authentication
- Profile Management
- Mentorship Request
- Mentorship Offer
- Chat Functionality
- Notifications

Junior & Senior Authentication

Using Firebase Authentication, the Authentication Module guarantees safe user registration, login, and account administration. A unique user ID is used to identify each person and associate data. It serves as the basis for user-specific functionality and offers email verification, password protection, and access restriction.

Profile Management Module

The Profile Management module allows users to create, update, and manage their personal profiles. It enables juniors to set up profiles for mentorship requests and seniors to set up profiles for offering mentorship. Users can edit their details, including contact information, and update their preferences.

Mentorship Request Module

The Mentorship Request module allows juniors to send mentorship requests to seniors based on specific areas of guidance. It enables users to specify their needs, making it easier for seniors to provide tailored advice. Notifications keep both parties informed about new requests.

Mentorship Offer Module

The Mentorship Offer module enables seniors to offer mentorship to juniors in specific areas of expertise. Seniors can view requests, accept or decline, and provide guidance through messages or meetings. This ensures a seamless connection between mentors and mentees.

Chat Functionality Module

The Chat Functionality module enables seamless communication between juniors and seniors. It allows users to send and receive text messages, facilitating real-time discussions. This module ensures a smooth interaction for mentorship and guidance, providing an essential communication tool within the app.

Notification Module

The Notification module is designed to keep users informed of important updates in real-time. It alerts users about new mentorship requests, messages, profile changes, and other relevant activities within the app. By providing timely notifications, it ensures that both juniors and seniors stay engaged and responsive, facilitating smooth communication and enhancing the overall user experience in the Grad-Chat application.

3.2 DATAFLOW DIAGRAM

The Data Flow Diagram provides information about the inputs and outputs of each entity and process itself.

LEVEL 0

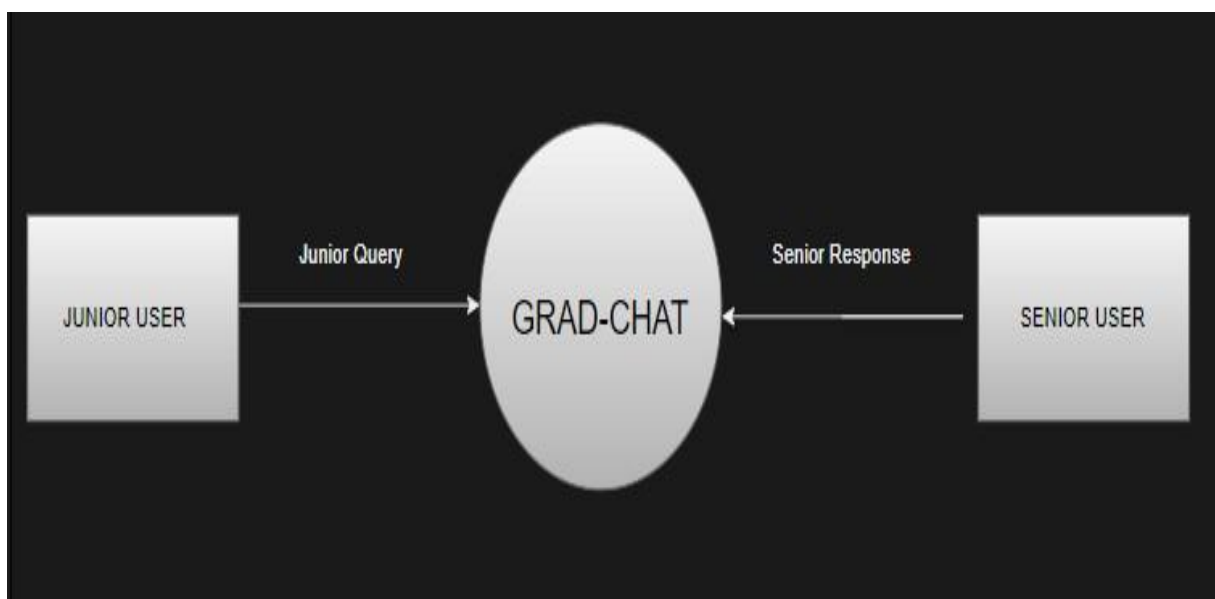


Figure 3.1
Dataflow Diagram Level 0

Figure 3.1, the "Grad-Chat" Level 0 Data Flow Diagram, illustrates the process where Junior Users send queries to the platform, which then forwards them to Senior Users for responses. Senior Users offer guidance and mentorship, enabling knowledge sharing and fostering a collaborative learning environment. This seamless communication ensures efficient support and engagement within the Grad-Chat community.

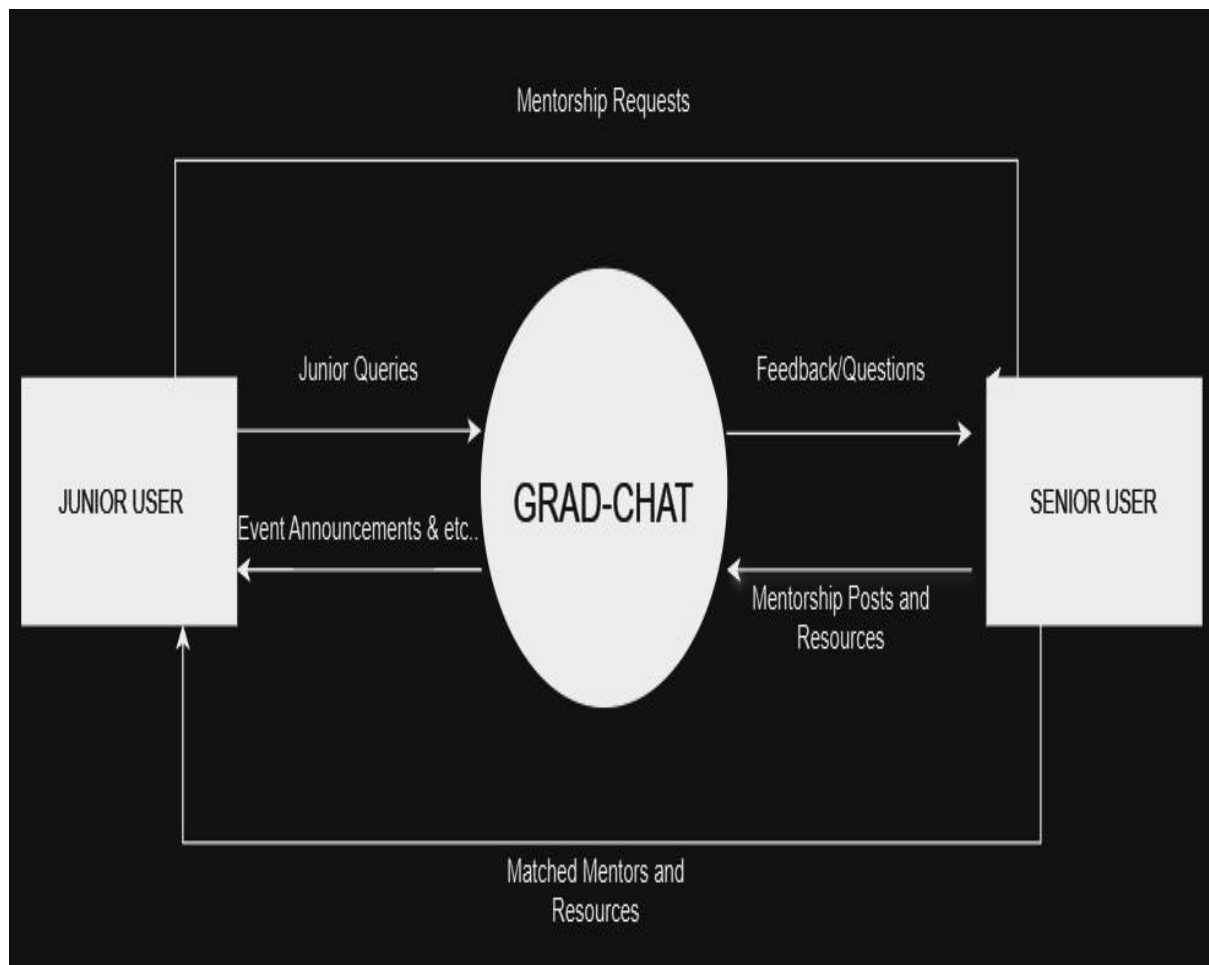
LEVEL 1

Figure 3.2
Dataflow Diagram Level 1

Figure 3.2, the "Grad-Chat" Level 1 Data Flow Diagram, provides a detailed overview of the interactions within the platform. Junior Users send queries and requests, which are matched to Senior Users based on expertise. Senior Users then share mentorship posts, offer feedback, and promote events. This dynamic exchange encourages effective knowledge-sharing, skill development, and fosters a supportive mentorship environment, helping to build a thriving learning community.

LEVEL 2

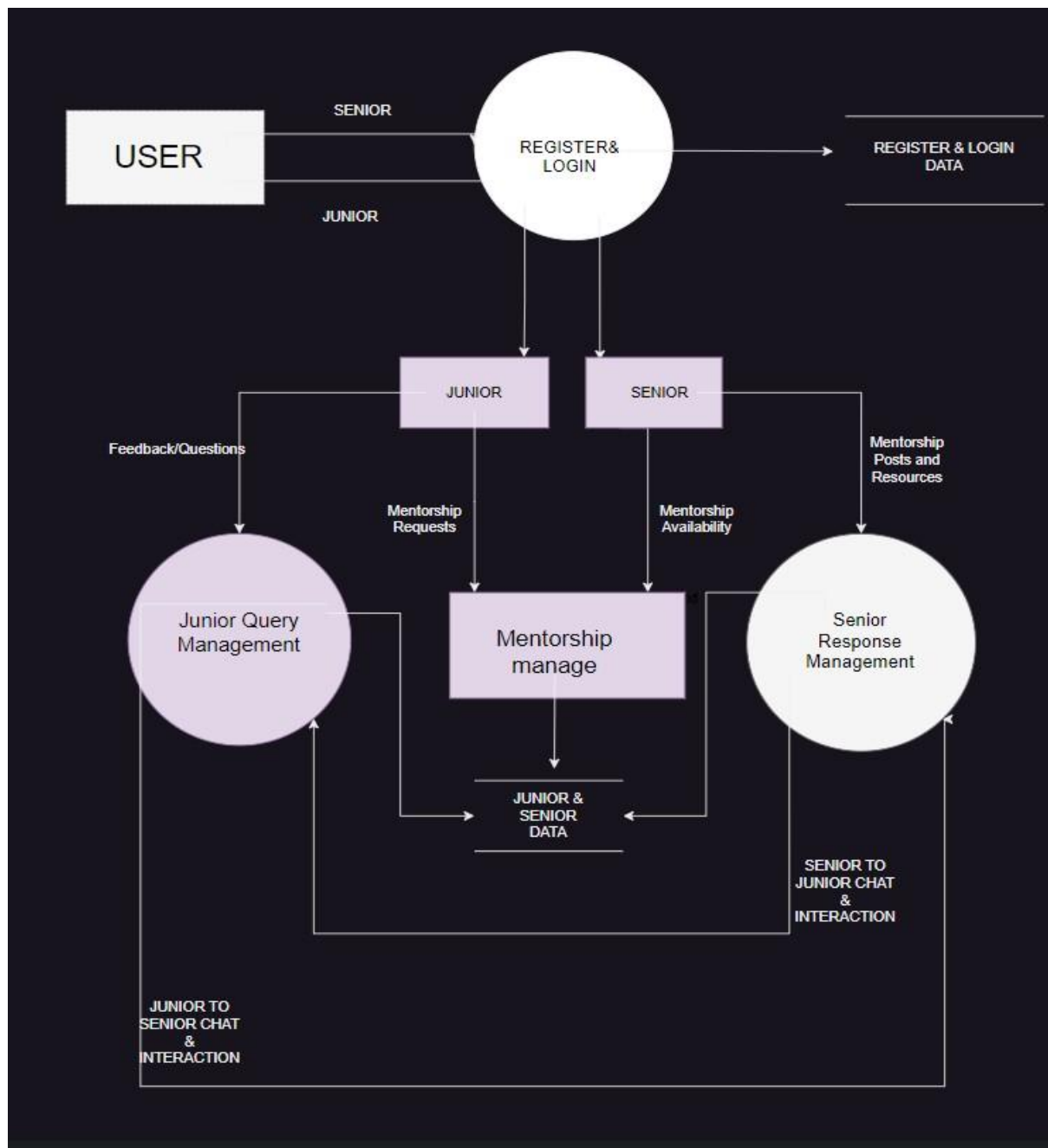


Figure 3.3
Dataflow Diagram Level 2

In Figure 3.3 "Grad-Chat" Level 2 DFD highlights core components: users register/login, juniors submit queries via Junior Query Management, and seniors provide responses via Senior Response Management. The Mentorship Manager connects them, enabling resource sharing, chats, and effective mentorship interactions.

User case diagram

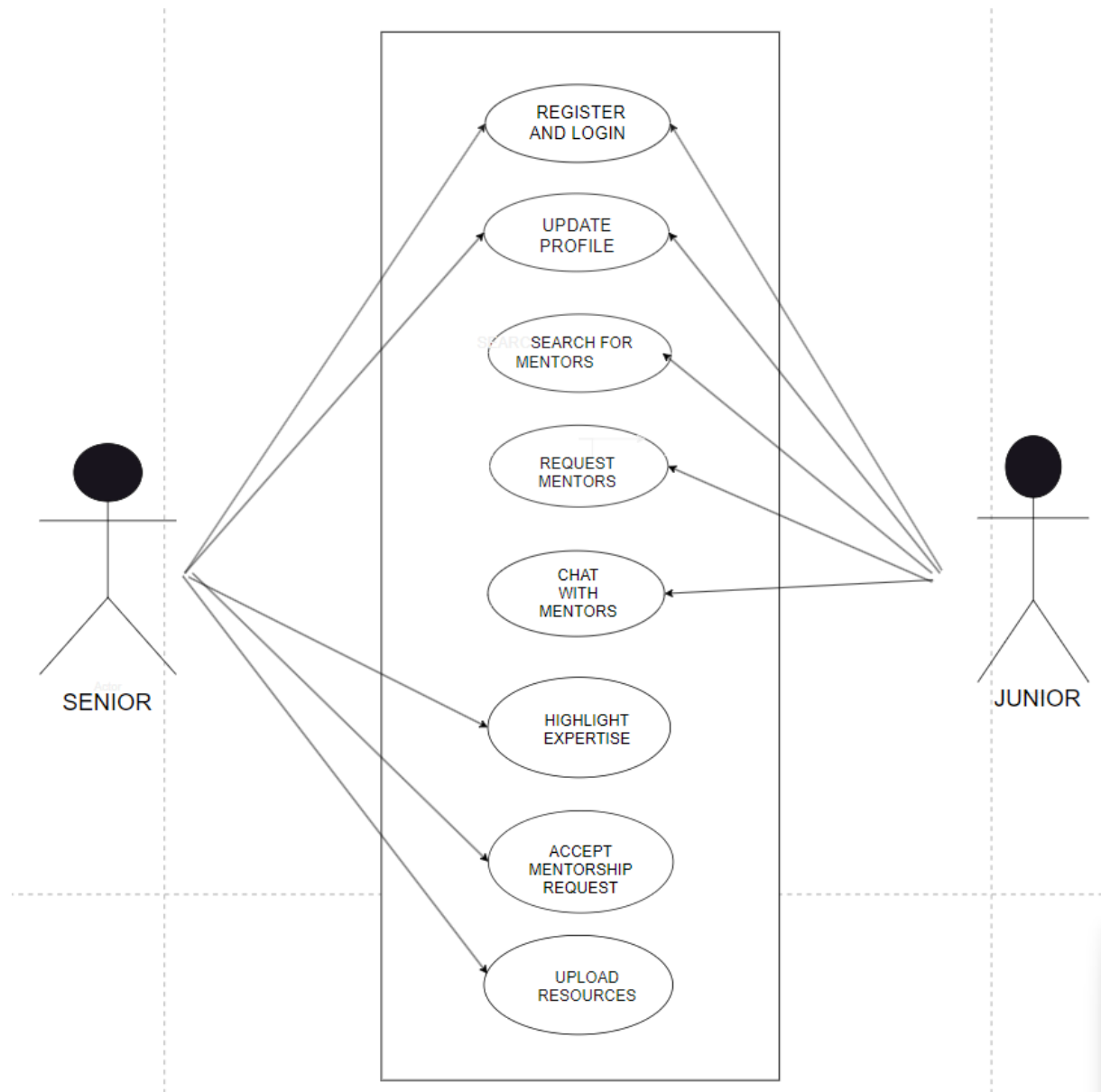


Figure 3.4
Use case diagram

Figure 3.3 illustrates a mentorship platform where seniors can register, update profiles, and highlight their expertise. Juniors can register, request mentors, and chat with them. Both parties can upload and share resources.

Figure 3.5

Class diagram

Figure 3.5 shows the Class diagram. The Grad-Chat project facilitates mentorship between college juniors and seniors, offering user-friendly features for guidance, resource sharing, private/public chats, and event coordination. The class diagram illustrates relationships among users, chats, resources, events, and notifications.

3.3 DATABASE DESIGN

Database design is a critical process that defines the structure, organization, and relationships of data within a database. It involves determining what data needs to be stored, how the data elements are interconnected, and how the data will be accessed and manipulated. The goal is to design an efficient system that ensures data integrity, easy retrieval, and minimal redundancy. Database design can be broken down into two main components: logical design and physical design. Logical design focuses on defining the structure of the database, including tables, relationships, keys, and normalization to ensure data consistency. Physical design deals with the actual implementation and optimization of the database system, such as indexing, partitioning, and tuning performance.

For object-oriented databases, entities and relationships are mapped directly to objects and classes, making it easier to manage complex data structures. In traditional relational databases, this process involves creating an Entity-Relationship (ER) diagram to represent data relationships. Additionally, database design also includes the creation of forms, queries, and reports that interact with the underlying data, offering an intuitive user interface for data management within the database management system (DBMS). The overall process ensures that the database serves its intended purpose efficiently and effectively.

Data Integration

Data integration involves combining data from various sources and presenting it as a unified system. While the data may be stored on different devices, it is logically centralized and connected through communication networks. This ensures data integrity by maintaining a single version of the data, making updates reflect across all applications accessing it. This approach reduces data redundancy by eliminating the need for multiple copies of the same data and minimizes storage requirements. By consolidating information, data integration ensures consistency, improves operational efficiency, and enhances decision-making by providing all users and applications with accurate and up-to-date data.

Data Independence

Data independence refers to the ability of application programs to remain unaffected by changes in the physical data structure or organization. It ensures that modifications made to the storage or organization of data do not require changes in the application code. This allows for greater flexibility in handling data, as database structure changes can occur without disrupting the operation of existing applications.

Data independence is crucial for the long-term maintenance and scalability of a system. It ensures that the logical structure of the data remains consistent even as physical storage or access methods evolve. For the Grad-Chat application, the database tables and their specifications were designed based on the system's requirements, ensuring that updates to the database structure will not require changes in the application code. This approach streamlines the process and ensures data consistency across various modules of the system.

Data Security

Data Security in Grad-Chat focuses on safeguarding user information and ensuring privacy throughout the app's lifecycle. As an application that involves the exchange of sensitive information, such as mentorship requests, personal profiles, and communication between juniors and seniors, protecting user data is critical. The application implements various security measures like encryption, secure password storage, and secure messaging to prevent unauthorized access and data corruption. User authentication is required for all accounts, ensuring that only authorized individuals can access the app's features. Role-based access control (RBAC) is employed to ensure that both junior and senior users can only access the data they are permitted to, further reducing the risk of unauthorized actions.

Additionally, the Grad-Chat app employs end-to-end encryption for chat functionality, ensuring that communication between users remains confidential. This means that messages and data shared in real-time are encrypted and can only be decrypted by the intended recipient. The app uses secure protocols to handle all data transactions,

including login and registration processes, ensuring that sensitive data like passwords and personal information are never exposed during transmission.

To protect user privacy, Grad-Chat ensures that data is stored in a secure, encrypted database. Regular security audits and updates are conducted to protect against emerging cyber threats. This comprehensive security strategy aims to prevent data breaches and ensure that the users' personal and sensitive data remains secure within the Grad-Chat ecosystem.

3.4 INPUT DESIGN

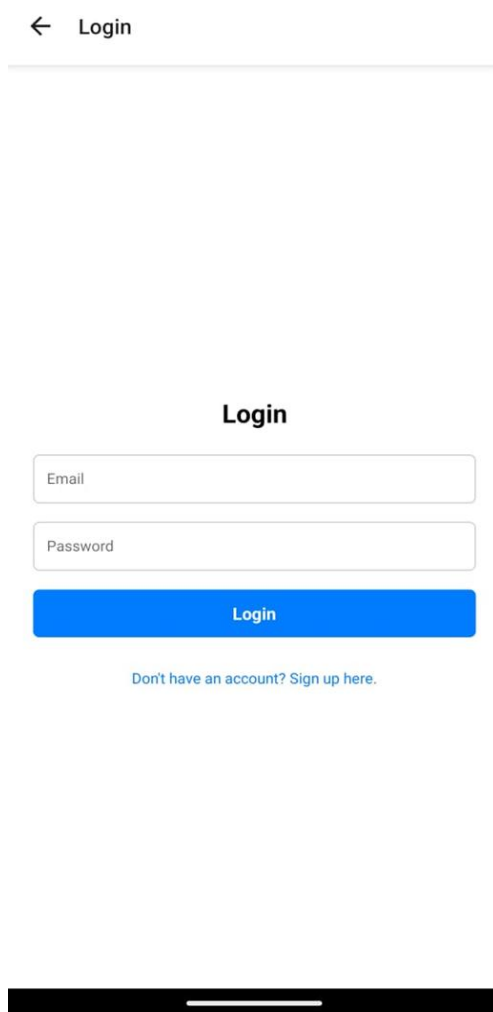
Input design in the Grad-Chat project focuses on ensuring a seamless user experience by converting user-originated inputs into a format that can be processed by the system efficiently. This phase is critical in the development of the application, as any issues related to user inputs can often lead to system failures or inefficiencies. The input design is carefully planned to ensure that all user inputs are correctly validated and processed in a way that prevents errors or inconsistencies.

To enhance user experience and reduce time consumption, the forms and input interfaces in Grad-Chat are designed to be intuitive and user-friendly. The system includes robust input validation techniques that ensure only correct and expected data are accepted. For instance, during the registration and login process, the application checks for valid email addresses, strong passwords, and proper formatting of user details.

Additionally, Grad-Chat minimizes the chances of error by providing clear prompts, tooltips, and input field constraints, which guide users to enter the correct data. By utilizing freely available technologies and incorporating these design practices, the project remains within budget while providing a smooth and efficient user interaction. This careful attention to input design contributes significantly to the overall performance and reliability of the application.

Login Screen

The user login form in Grad-Chat allows users to access the application using their registered email ID and password. Users must provide the correct registration details to log in; otherwise, access will be denied, ensuring secure authentication.

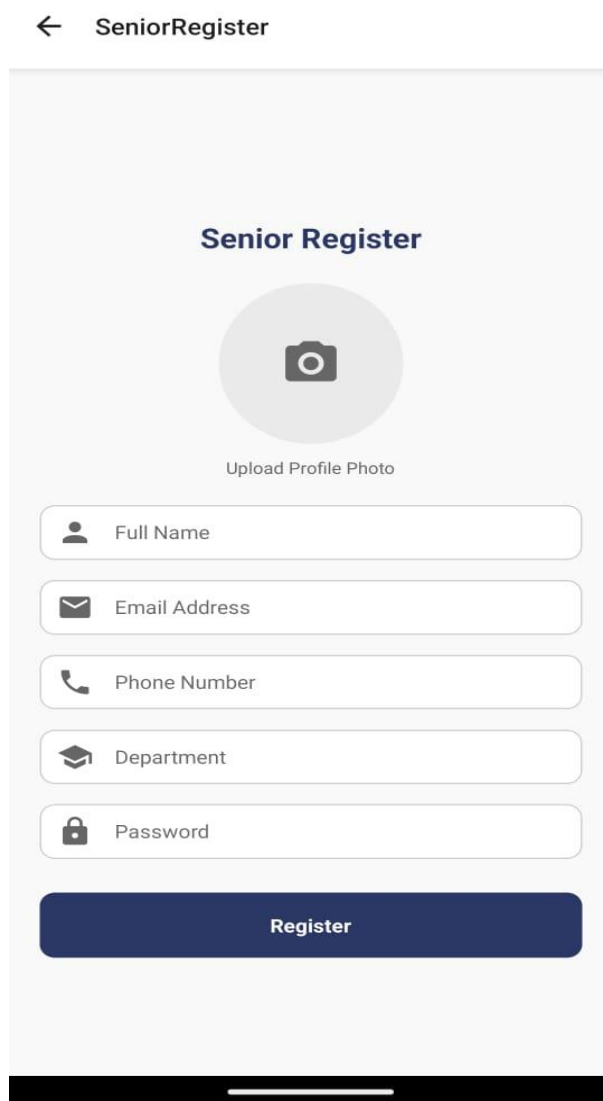


The image shows a mobile application login screen. At the top, there is a back arrow icon followed by the text "Login". Below this is a horizontal separator line. In the center, the word "Login" is displayed in bold. Underneath, there are two input fields: the first is labeled "Email" and the second is labeled "Password". Below these fields is a blue button with the text "Login". At the bottom of the form area, there is a link that says "Don't have an account? Sign up here." The entire screen is framed by a black bar at the bottom, which contains a white horizontal line, likely representing a mobile home indicator.

Figure 3.5 Login Screen

The Figure 3.4 is user login form. Login form has mail id and password. The username should be in proper email format. If the fields are not filled, form will not be submitted. Password must contain at least 8 digit or characters. If the form is submitted successfully the page redirected towards home page.

Senior Register Screen

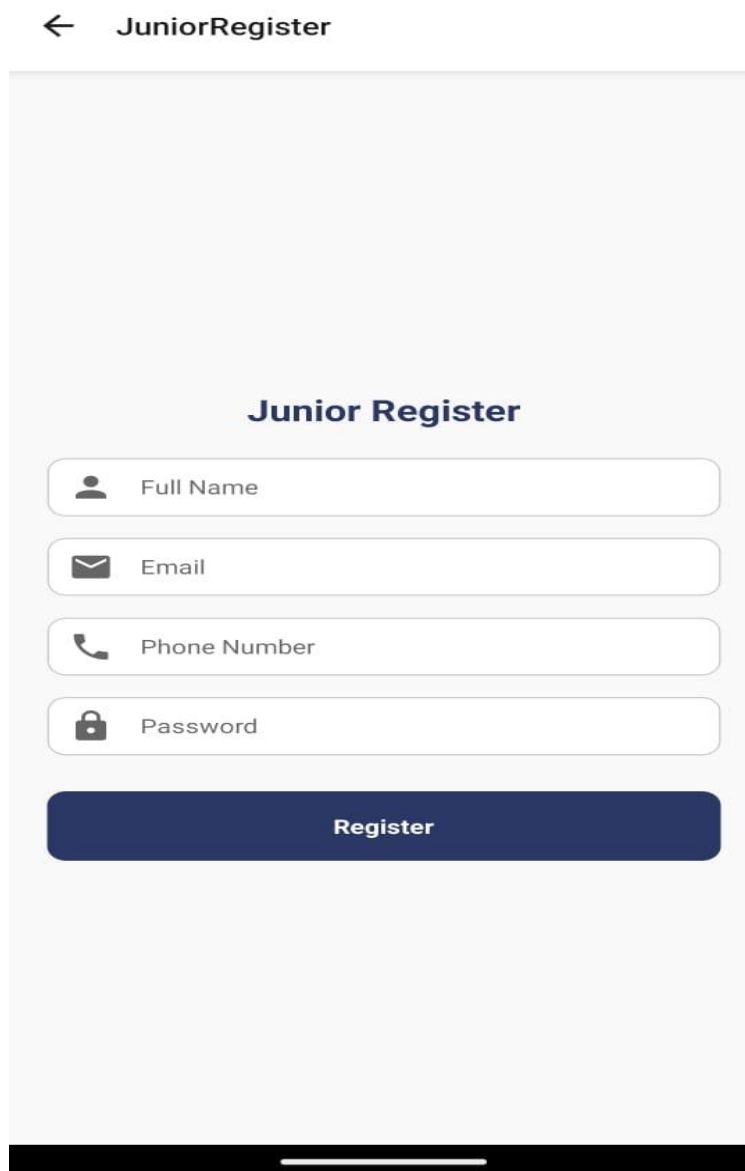


The image shows a mobile app screen titled "Senior Register". At the top, there is a back arrow and the text "SeniorRegister". Below this, the title "Senior Register" is centered. Under the title is a circular placeholder for a profile photo with a camera icon and the text "Upload Profile Photo". Below the photo placeholder are five input fields, each with an icon and a label: "Full Name" (person icon), "Email Address" (envelope icon), "Phone Number" (phone icon), "Department" (graduation cap icon), and "Password" (lock icon). At the bottom of the form is a dark blue button labeled "Register". The screen is set against a light gray background with rounded corners.

Figure 3.5 Senior Register Screen

The Senior Registration screen enables seniors to create a profile with details like name, department, and areas of expertise. This allows them to offer mentorship in specific categories such as placements, academics, and event management.

Junior Register Screen



The image shows a mobile application screen titled "Junior Register". At the top, there is a back arrow and the text "JuniorRegister". The main content area has a light gray background. In the center, the title "Junior Register" is displayed in a bold, dark blue font. Below the title, there are four input fields, each with a small icon on the left and a label on the right: "Full Name" (person icon), "Email" (envelope icon), "Phone Number" (phone handset icon), and "Password" (lock icon). All input fields are white with rounded corners and a thin gray border. Below these fields is a large, dark blue button with the word "Register" in white, bold text. At the very bottom of the screen, there is a black bar with a white horizontal line in the center, representing the mobile home indicator.

Figure 3.6 Junior Register Screen

The Junior Registration screen allows juniors to create a profile by entering details such as name, department, and areas of interest. This enables them to seek mentorship in areas like academics, placements, and personal development, helping them connect with senior mentors for guidance.

3.5 OUTPUT DESIGN

Output design refers to the presentation of information generated by the system for the end users. It focuses on creating results that are not only clear and understandable but also presented in an aesthetically pleasing format. As the primary source of information for users, effective output design is essential for user satisfaction. It encompasses the design of reports, screens, and other forms of data display, ensuring the information is easily interpretable.

The design process involves identifying the specific output needed to meet user requirements and ensuring that it is both functional and user-friendly. User feedback on output is critical, as they are the ultimate judges of its quality and usability. In the case of this project, the output is designed to be informative, convenient, and visually appealing, with a focus on enhancing the user experience. Efficient output design helps users make better decisions and interact effectively with the system.

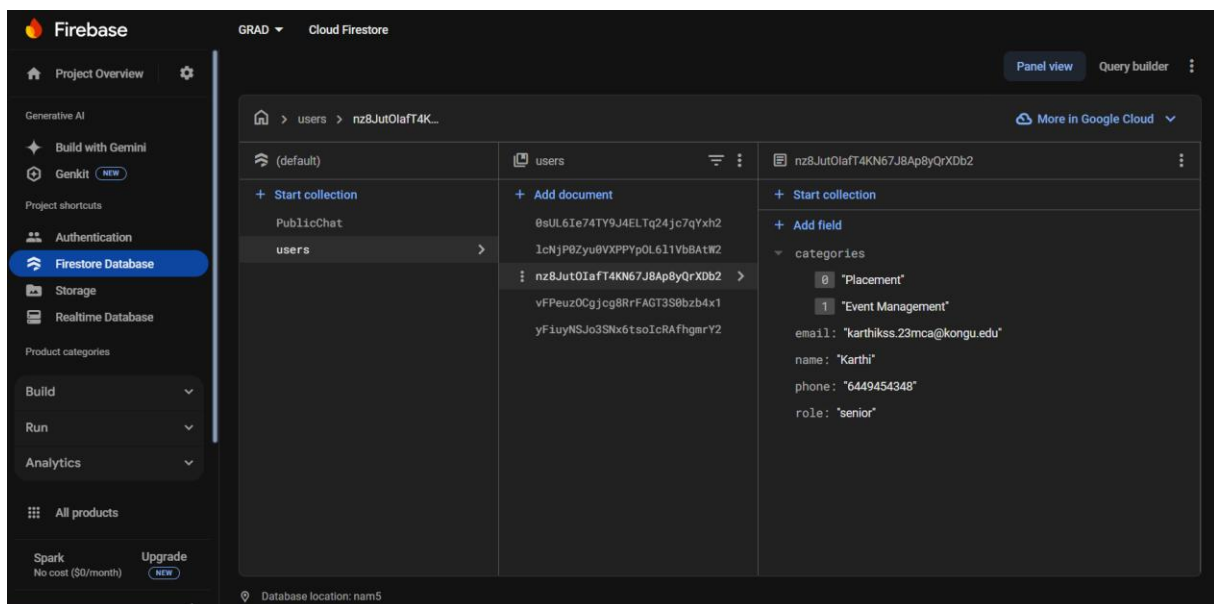


Figure 3.7 Storing in Firebase

The Figure 3.6 shows the information like details of login, user details. In Firebase data is stored as JSON application data.

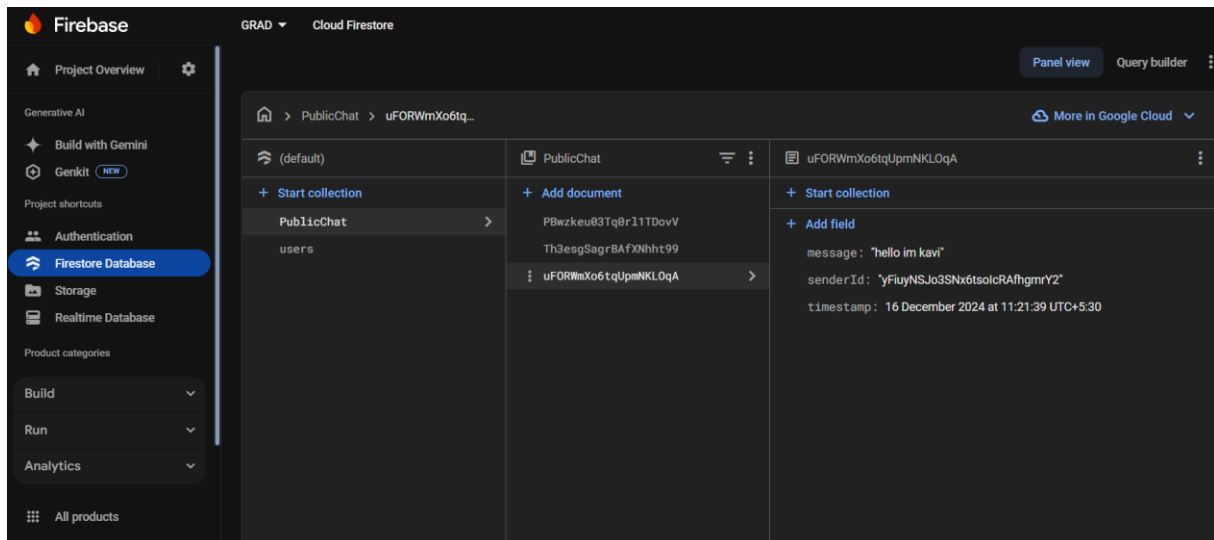


Figure 3.7 Community Chat Stored In Firebase

The Figure 3.7 Shows the information that stores in Firebase about Community Chat.

SUMMARY

This chapter provides a detailed description of the modules in the "Grad-Chat" application, including Profile Management, Mentorship Request, Mentorship Offer, Chat Functionality, and Notification. The Data Flow Diagrams (DFD) illustrate the application's process flow, starting from user input and progressing through various processes until user exit.

CHAPTER 4

IMPLEMENTATION

4.1 CODE DESCRIPTION

The code description provides an overview of the project's implementation, explaining the purpose and structure of the code. It focuses on clarifying the intent behind the code, rather than simply restating its functionality. Flutter is used as the front-end framework to build a responsive user interface, while Firebase is utilized as the back-end solution for real-time data storage and authentication. Clear and concise comments have been added to the code for better understanding and maintainability.

4.2 STANDARDIZATION OF THE CODING

Coding standards are essential guidelines that define the programming style and structure, ensuring consistency, readability, and maintainability of the code. These standards provide a set of rules for formatting, naming conventions, and comment usage. Following coding standards from the beginning of the project makes it easier to maintain and debug the code in the long run. In this project, all coding standards have been strictly adhered to, ensuring that the code is clean, organized, and easy to understand. This includes consistent naming conventions for variables and functions, proper indentation, and clear, concise comments to explain the code's logic. Standardizing the coding process ensures that all team members can work cohesively and that the code is scalable and maintainable for future development. By establishing and following these coding practices, the project remains efficient and easily modifiable over time, ensuring a smoother development process and reducing the potential for errors.

4.3 ERROR HANDLING

Error handling is a critical aspect of software development that involves managing unexpected or abnormal events that occur during the execution of a program. It ensures that the application can gracefully handle errors without crashing, providing a smooth

user experience. In this project, error handling is implemented using try-catch blocks in Flutter, which allows the app to catch and handle exceptions. Common errors include failure to fetch data from Firebase, invalid user inputs, or network connectivity issues. When an error occurs, appropriate messages are displayed to the user, offering suggestions to resolve the issue and continue using the application. This ensures that users are informed and can take corrective actions. Moreover, in cases of severe errors, the app logs the error and sends it to the developer for investigation. To prevent errors in production, error monitoring tools and bug tracking systems are used, allowing the development team to identify and fix issues promptly. By implementing proper error handling, the project ensures reliability and robustness, minimizing downtime and providing a positive user experience.

USER INTERFACE DESIGN

Input design is a crucial phase in the development of a computerized system, focusing on converting user inputs into a format that can be understood and processed by the computer. It involves careful planning and analysis to ensure that input data is accurate, easy to use, and cost-effective. Effective input design helps prevent issues that can arise from incorrect or inefficient data entry, which can lead to system failures. This process requires analyzing the nature of the data to be input, selecting the appropriate medium for input (such as text fields, buttons, or dropdowns), and determining how data will be organized, coded, and validated. It also involves creating intuitive dialogues and forms that guide the user in providing the necessary information. The input data may not always be raw data captured from scratch but could also be sourced from other systems or subsystems. Proper input design minimizes errors, ensures user understanding, and helps in achieving the highest accuracy possible. In this project, input design is focused on simplicity and accuracy, with clear validation mechanisms to ensure that only valid data is entered into the system.

SUMMARY

This chapter provides an overview of the code description, emphasizing that its purpose is to summarize the code and clarify the programmer's intent, without repeating or explaining the code. It highlights the importance of coding standards, which define a programming style and ensure maintainable code. Additionally, the chapter discusses exception handling as a process for managing errors in the code. The next chapter, Chapter 5, will focus on testing procedures and strategies for the project.

CHAPTER 5

TESTING AND RESULTS

5.1 TESTING

Software testing serves as the final assessment of specifications, designs, and coding and is a crucial component of software quality assurance. The system is tested throughout the testing phase utilizing diverse test data. The preparation of test data is essential to the system testing process. The system under study is tested after the test data preparation. Once the source code is complete, relevant data structures should be documented. The finished project must go through testing and validation, when errors are explicitly targeted and attempted to be found.

The project developer is always in charge of testing each of the program's separate units, or modules. Developers frequently also perform integration testing, which is the testing phase that results in the creation of the complete program structure.

This project has undergone the following testing procedures to ensure its correctness

- Unit testing
- Integration Testing
- Validation Testing

5.1.1 Unit Testing

Unit testing was performed to verify that individual components of the system work as expected. The testing focused on the specific functionality of each module and ensured that the modules perform correctly in isolation. During the development phase, unit testing helped identify coding errors and logical issues within individual modules.

Test Case 1

Module : User Login

Input : Username and Password

Event : Button click

Output : Logged in successfully

Test 1

Module : User Login

Username : abc@gmail.com

Password : 123456

Output : Logged in successfully

Event : Button click

Analysis : Username and Password has been verified

Test 2

Module : Senior Register

Input : Name,Department,Expertise

Output : Profile Saved successfully

Event : Button click

Analysis : Username and Password has been checked and error shown

5.1.2 Integration Testing

Integration testing ensures that different modules of the system work together as a cohesive unit. It verifies the interactions between integrated modules, ensuring they function correctly when combined. Test data is generated automatically to simulate real-world interactions and detect any issues in the interfaces, confirming that the integrated modules work seamlessly together and deliver the expected output, such as proper user login, profile registration, and mentorship request processes, without errors or disruptions.

Test Case 1

Module	: User Login
Input	: Username and Password
Output	: Redirect to Profile Screen

Test 1

Module	: User Login
Username	: abc@gmail.com
Password	: 123456
Output	: Redirected to Profile Screen
Analysis	: Username and Password has been verified

Test 2

Module	: User Login
Username	: abcgmail.com
Password	: 123456
Output	: Enter the correct username
Analysis	: Username and Password has been checked and error shown

5.1.3 Validation Testing

Validation testing ensures that the system meets customer requirements and functions as intended before delivery. This process verifies that all functionalities are correctly implemented and satisfy the specified criteria. It ensures that the software works as expected and meets user expectations, confirming that it is ready for deployment.

Test case 1

Module	: User Registration
Input	: Password
Output	: Password must be at least 8 numbers or characters

Test 1

Module	: User Registration
Password	: 123456
Output	: Registered successfully
Analysis	: The given password is 8 numbers. So, it is validated.

Test 2

Module	: User Registration
Password	: 1234
Output	: Password must be at least 8 characters or number
Analysis	: Only 6 digit numbers or characters should be validated.

SUMMARY

The previous chapter covers various testing types, including unit testing, integration testing, and validation testing. After the completion of the coding phase, the system undergoes thorough evaluation and documentation, focusing on data structures and functionality. The final product is tested and validated to ensure it meets user requirements and functions correctly, with attention to both minor and major issues.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The "Grad-Chat" platform is designed to foster mentorship and communication between juniors and seniors in a college environment. This project provides a seamless experience for junior users to seek guidance from senior mentors on various topics, including academics, placements, and event management. It offers features like personalized mentorship requests, profiles, chat functionality, and notifications to ensure smooth communication.

In addition to these core features, the inclusion of an AI chatbot further enhances the user experience by offering quick, automated responses to common questions, ensuring that juniors have access to instant support when needed. The platform also integrates secure login, user registration, and profile management to safeguard user data and ensure a trustworthy environment.

The system's design focuses on simplicity and user-friendliness, allowing easy navigation and interaction between users. By promoting mentorship, knowledge-sharing, and a supportive community, "Grad-Chat" helps build stronger connections within the college environment, contributing to personal and academic growth.

6.2 FUTURE ENHANCEMENT

- Enhance the AI chatbot by improving its functionality to provide more personalized mentoring advice and answer specific questions related to academic progress, placements, and events.
- Add an event calendar feature to allow juniors and seniors to stay updated on academic events, workshops, and placement drives.
- Enable the sharing of resource materials such as books, study guides, and placement preparation tips between juniors and seniors.
- Integrate a notification system for real-time updates about new mentorship

opportunities, session reminders, and upcoming events.

- Implement a user feedback system for juniors to rate their mentorship experience, helping to improve the matching process and overall quality of mentorship.
- Expand the profile features to include academic achievements, extracurricular activities, and personalized interests to help juniors better connect with seniors.

APPENDICES

A.SAMPLE CODING

LOGIN SCREEN CODE

```
import React, { useState } from "react"; import { View, Text, TextInput, TouchableOpacity,
StyleSheet, Alert } from "react-native"; import { auth } from "../firebaseConfig"; // Ensure
Firebase is configured in this file import { signInWithEmailAndPassword } from
"firebase/auth";
```

```
const Login = ({ navigation }) => {   const [email, setEmail] = useState("");   const
[password, setPassword] = useState("");   const [loading, setLoading] = useState(false);
```

```
const validateAndLogin = async () => {
  if (!email || !password) {
    Alert.alert("Error", "All fields are required!");      return;
  }
```

```
const emailRegex = /^[a-zA-Z]+\.(24mca)@kongu\.edu$/;
```

```
if (!emailRegex.test(email)) {
  Alert.alert(      "Error",
    "Please enter a valid email address in the format: name.24mca@kongu.edu"
  );      return;
}
```

```
if (password.length < 6) {
  Alert.alert("Error", "Password must be at least 6 characters long!");      return;
}
```

```

        setLoading(true);
    try {
        await signInWithEmailAndPassword(auth, email, password);
        Alert.alert("Success", "You are logged in!");
    } catch (error) {
        console.error("Error logging in: ", error.message);
        Alert.alert("Login Failed", error.message);
    } finally {
        setLoading(false);
    }
};

return (
    <View style={styles.container}>
        <Text style={styles.title}>Login</Text>

        <TextInput
            style={styles.input}
            placeholder="Email"
            keyboardType="email-address"
            value={email}
            onChangeText={setEmail}
            autoCapitalize="none"
        />

        <TextInput
            style={styles.input}
            placeholder="Password"
            secureTextEntry
            value={password}
            onChangeText={setPassword}
        />

        <TouchableOpacity
            style={styles.button}
            onPress={validateAndLogin}
            disabled={loading}
        >
            <Text style={styles.buttonText}>{loading ? "Logging In..." : "Login"}</Text>
        </TouchableOpacity>
    </View>
);

```

```

    <TouchableOpacity
      onPress={() => navigation.navigate("Signup")}          style={styles.link}
    >
    <Text style={styles.linkText}>Don't have an account? Sign up here.</Text>
  </TouchableOpacity>
</View>
);
};

```

```

const styles = StyleSheet.create({
  container: {    flex: 1,    padding: 20,    justifyContent: "center",
    backgroundColor: "#fff",
  },  title: {    fontSize: 24,    fontWeight: "bold",    marginBottom: 20,
    textAlign: "center",
  },  input: {    height: 40,    borderColor: "#ccc",    borderWidth: 1,
borderRadius: 5,    marginBottom: 15,    paddingHorizontal: 10,
  },
  button: {
    backgroundColor: "#007BFF",
    padding: 10,    borderRadius: 5,    alignItems: "center",
    marginBottom: 15,
  },
  buttonText: {    color: "#fff",    fontSize: 16,    fontWeight: "bold",
  },  link: {    marginTop: 10,
    alignItems: "center",
  },
  linkText: {    color: "#007BFF",    fontSize: 14,
  },
});

```

```

export default Login;

```

App.js code

```
import React, { useState, useEffect } from 'react'; import { View, ActivityIndicator,
StyleSheet } from 'react-native'; import { NavigationContainer } from '@react-
navigation/native'; import { createStackNavigator } from '@react-navigation/stack'; import {
createBottomTabNavigator } from '@react-navigation/bottom-tabs'; import {
onAuthStateChanged, getAuth } from 'firebase/auth'; import Icon from 'react-native-vector-
icons/Ionicons'; // For tab icons
import Welcome from './screens/Welcome'; import Login from './screens/Login'; import
SeniorRegister from './screens/SeniorRegister'; import JuniorRegister from
'./screens/JuniorRegister'; import Home from './screens/Home';
import CommunityChat from './screens/CommunityChat';
import Profile from './screens/Profile'; import ChatBot from './screens/ChatBot';
import './firebaseConfig'; // Ensure Firebase is initialized in this file import PlacementUser
from './screens/PlacementUser';
import PlacementUserDetail from './screens/PlacementUserDetail';

const Stack = createStackNavigator();
const Tab = createBottomTabNavigator();

const App = () => {
  const [isAuthenticated, setIsAuthenticated] = useState(null); // null indicates loading state
  const auth = getAuth();

  useEffect(() => {    const unsubscribe = onAuthStateChanged(auth, (user) => {
setIsAuthenticated(!!user); // If user exists, set to true; else false    });

    return () => unsubscribe(); // Cleanup subscription on unmount  }, [auth]);

  if (isAuthenticated === null) {
// Show a loading indicator while checking authentication state return (
<View style={styles.loaderContainer}>
```



```

<ActivityIndicator size="large" color="#0000ff" />
</View>
);
}

// Tab Navigator for authenticated users  const TabNavigator = () => (
screenOptions=(({ route }) => ({      tabBarIcon: ({ color, size }) => {      let iconName;

        switch (route.name) {          case 'Home':
            iconName = 'home-outline';          break;          case 'CommunityChat':
iconName = 'chatbubbles-outline';          break;          case 'Profile':
            iconName = 'person-outline';
            break;          case 'ChatBot':
            iconName = 'chatbox-ellipses-outline';          break;          default:
            iconName = 'home-outline';
        }

        return <Icon name={iconName} size={size} color={color} />;
    },    )})
tabBarOptions={ {
    tabBarActiveTintColor: '#2A3663', // Active icon color    tabBarInactiveTintColor:
'gray',
    style: { height: 60, paddingBottom: 10 },
    }}
>
    <Tab.Screen name="Home" component={Home} />
    <Tab.Screen name="CommunityChat" component={CommunityChat} />
    <Tab.Screen name="Profile" component={Profile} />
    <Tab.Screen name="ChatBot" component={ChatBot} />
</Tab.Navigator>
);

```

```

return (
  <NavigationContainer>
    <Stack.Navigator>
      {isAuthenticated ? (
        <>
          <Stack.Screen      name="Main"      component={TabNavigator}
            options={{ headerShown: false }}
          />
          <Stack.Screen name="PlacementUser" component={PlacementUser} />
          <Stack.Screen name="PlacementUserDetail" component={PlacementUserDetail} />
        </>
      ) : (
        <>
          <Stack.Screen name="Welcome" component={Welcome} options={{ headerShown:
false }} />
          <Stack.Screen name="Login" component={Login} />
          <Stack.Screen name="JuniorRegister" component={JuniorRegister} />
          <Stack.Screen name="SeniorRegister" component={SeniorRegister} />
        </>
      )}
    </Stack.Navigator>
  </NavigationContainer>
);
};

const styles = StyleSheet.create({ loaderContainer: { flex: 1,  justifyContent: 'center',
alignItems: 'center',
  backgroundColor: '#fff',
}, });

```

export default App;

homescreen code

```
// Import required dependencies import React from 'react';
import { StyleSheet, View, Text, FlatList, TouchableOpacity, Image, ScrollView } from
'reactnative';

const categories = [
  {
id: 1,
name: 'Placement Volunteers', nav: 'PlacementUser',
//image: 'https://media.istockphoto.com/id/1371896330/photo/happy-asian-woman-in-
hisgraduation-day.jpg?s=612x612&w=0&k=20&c=Ur3moW11fKFms-
6UACseglMjoYAynYKzsanZpgK8lFk=',
description: 'Get guidance on placement preparation and strategies.'
}, {
id: 2,
name: 'Placed Super Seniors',
nav: "",
//image: 'https://media.istockphoto.com/id/1371896330/photo/happy-asian-woman-in-
hisgraduation-day.jpg?s=612x612&w=0&k=20&c=Ur3moW11fKFms-
6UACseglMjoYAynYKzsanZpgK8lFk=',
description: 'Learn from experiences of placed seniors.'
}, {
id: 3,
name: 'Event Coordinators',
nav: "",
// image: 'https://media.istockphoto.com/id/1371896330/photo/happy-asian-woman-in-
hisgraduation-day.jpg?s=612x612&w=0&k=20&c=Ur3moW11fKFms-
6UACseglMjoYAynYKzsanZpgK8lFk=',
description: 'Organize and manage college events effectively.'
}, {
id: 4, name: 'Toppers',
nav: "",
```

```

    //image: 'https://media.istockphoto.com/id/1371896330/photo/happy-asian-woman-in-
hisgraduation-day.jpg?s=612x612&w=0&k=20&c=Ur3moWl1fKFms-
6UACseglMjoYAynYKzsanZpgK8lFk=',
    description: 'Get study tips and guidance from department toppers.'
  }
];

```

```

const CategoryCard = ({ category, onPress }) => (
  <TouchableOpacity style={styles.card} onPress={() => onPress(category.name)}>
    <Image source={{ uri: category.image }} style={styles.cardImage}
      onError={() => console.warn(`Image failed to load for: ${category.name}`)}
    />
    <Text style={styles.cardTitle}>{category.name}</Text>
    <Text style={styles.cardDescription}>{category.description}</Text>
  </TouchableOpacity>
);

```

```

const HomePage = ({ navigation }) => {
  return (
    <View style={styles.container}>
      <View style={styles.headerContainer}>
        <Text style={styles.headerTitle}>Welcome to MentorConnect</Text>
        <Text style={styles.headerSubtitle}>Find the right mentors to guide your
journey</Text>      </View>

      <FlatList data={categories}
        keyExtractor={(item) => item.id.toString()} renderItem={({ item }) => (
          <CategoryCard category={item} onPress={() => navigation.navigate(item.nav)} />
        )}
        contentContainerStyle={styles.listContainer}
      />
    </View>
  );
};

```

```

);
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f5f5f5'
  },
  headerContainer: {
    backgroundColor: '#4CAF50', padding: 20,
    borderBottomLeftRadius: 20, borderBottomRightRadius: 20,
    alignItems: 'center'
  }, headerTitle: {
    color: 'fff', fontSize: 24,
    fontWeight: 'bold'
  },
  headerSubtitle: {
    color: 'fff', fontSize: 16, marginTop: 10, textAlign: 'center'
  },
  listContainer: {
    padding: 20
  }, card: {
    backgroundColor: 'fff', marginBottom: 15, padding: 15, borderRadius:
10, shadowColor: '#000', shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1, shadowRadius: 5,
    elevation: 3
  },
  cardImage: {
    width: '100%', height: 120, borderRadius: 10,
    marginBottom: 10
  }, cardTitle: {
    fontSize: 18, fontWeight: 'bold', color: '#333'
  },
  cardDescription: {
    fontSize: 14, .}
});

export default Home;

```

B.SCREENSHOTS

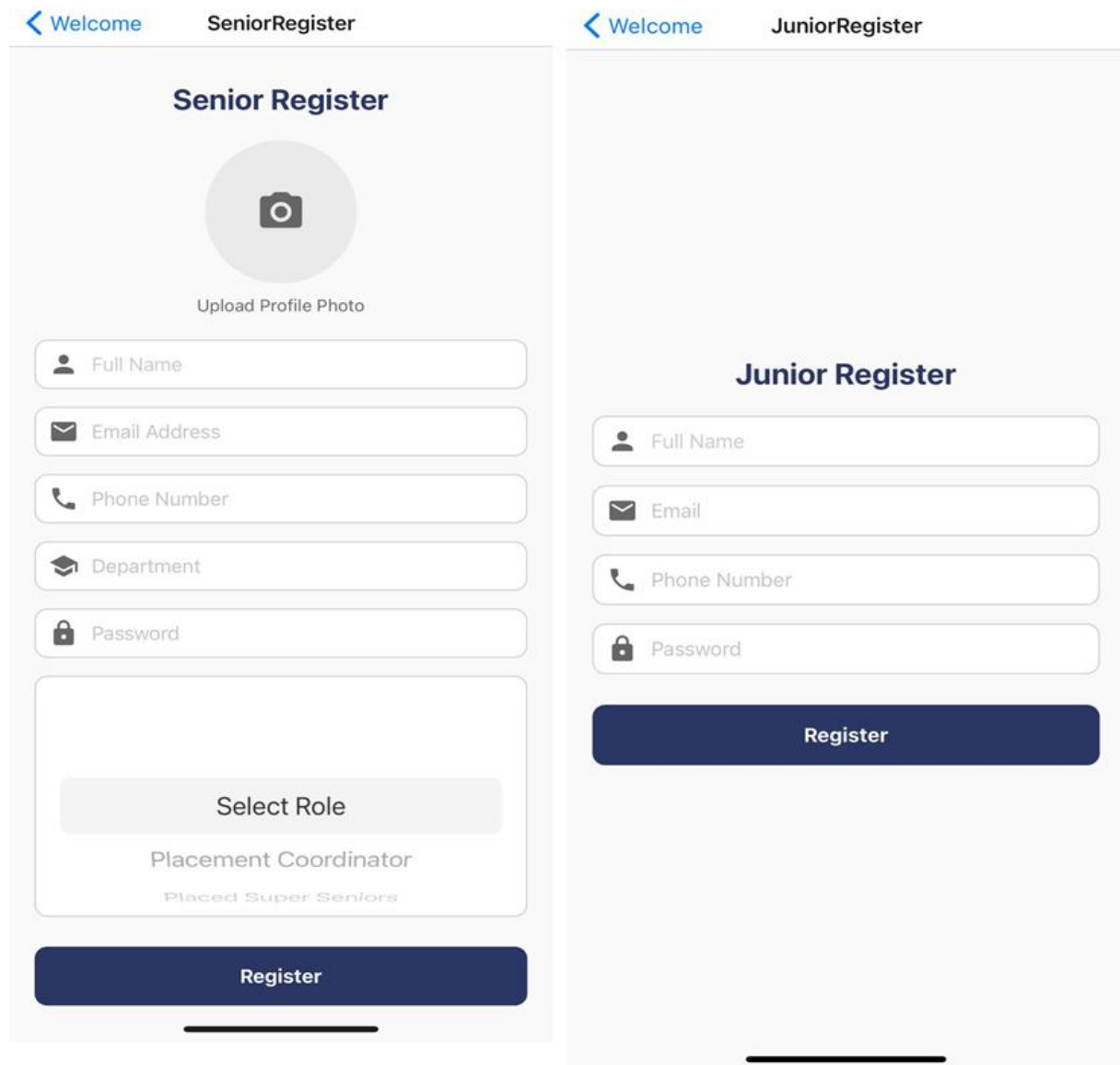
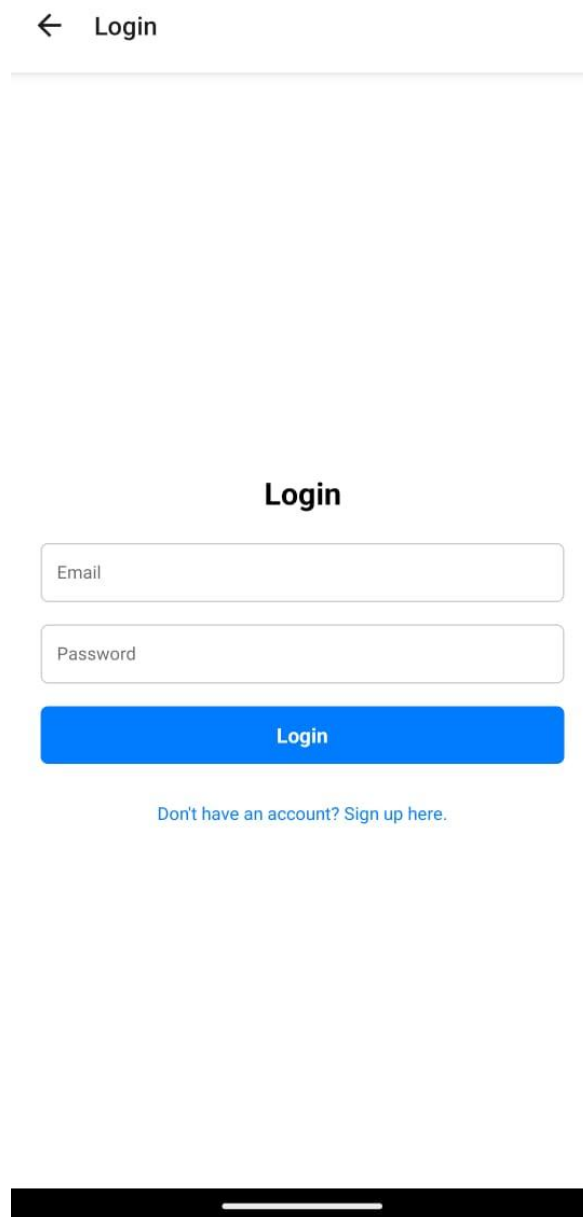


Figure B.1

Senior & Junior Register Screen

The Junior Registration screen enables juniors to provide their details for mentorship, while the Senior Registration screen allows seniors to share expertise, fostering guidance in areas like academics, placements, and event management for effective collaboration.



The image shows a mobile application login screen. At the top left, there is a back arrow icon followed by the text 'Login'. Below this is a horizontal separator line. In the center of the screen, the word 'Login' is displayed in a bold font. Underneath, there are two input fields: the first is labeled 'Email' and the second is labeled 'Password'. Below these fields is a prominent blue button with the text 'Login' in white. Under the button, there is a link that reads 'Don't have an account? Sign up here.' At the very bottom of the screen, there is a black horizontal bar with a white line in the center, representing a mobile home indicator.

Figure B.2
login Screen

The Login screen allows registered users, both juniors and seniors, to access their profiles by entering their username and password. It provides a secure way to manage user authentication and access the platform.



Figure B.3

Community Chat Screen

The CommunityChat screen allows juniors and seniors to engage in real-time conversations, share experiences, and discuss various topics. It fosters a collaborative environment for mentorship and community building.

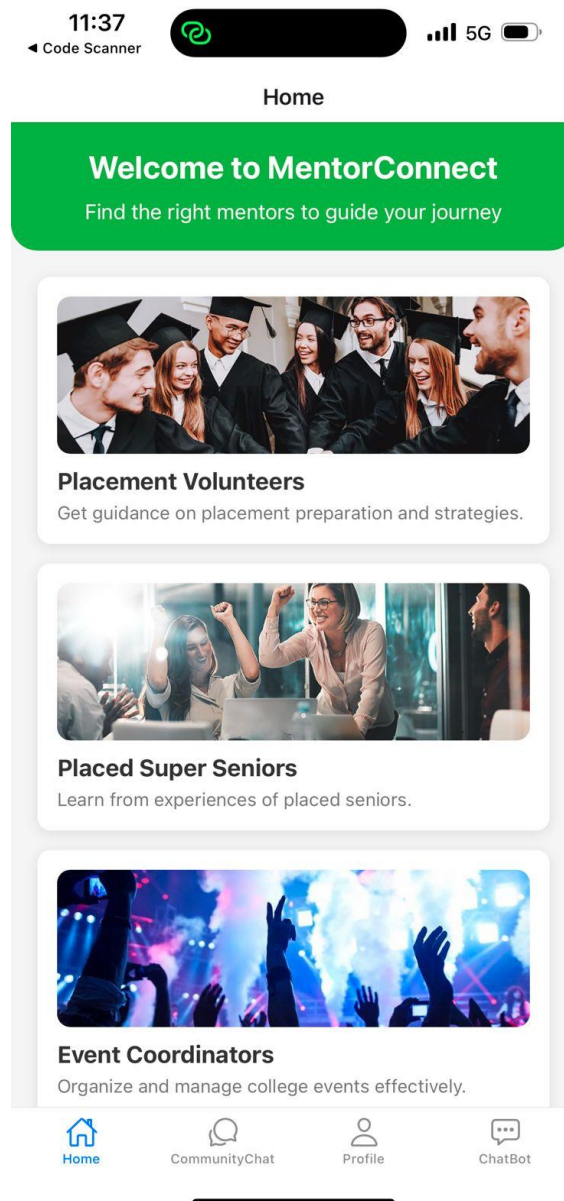


Figure B.4

Home Screen

The Home screen serves as the central hub, providing quick access to features like mentor profiles, community chats, AI ChatBot, and real-time updates. It ensures seamless navigation and user engagement

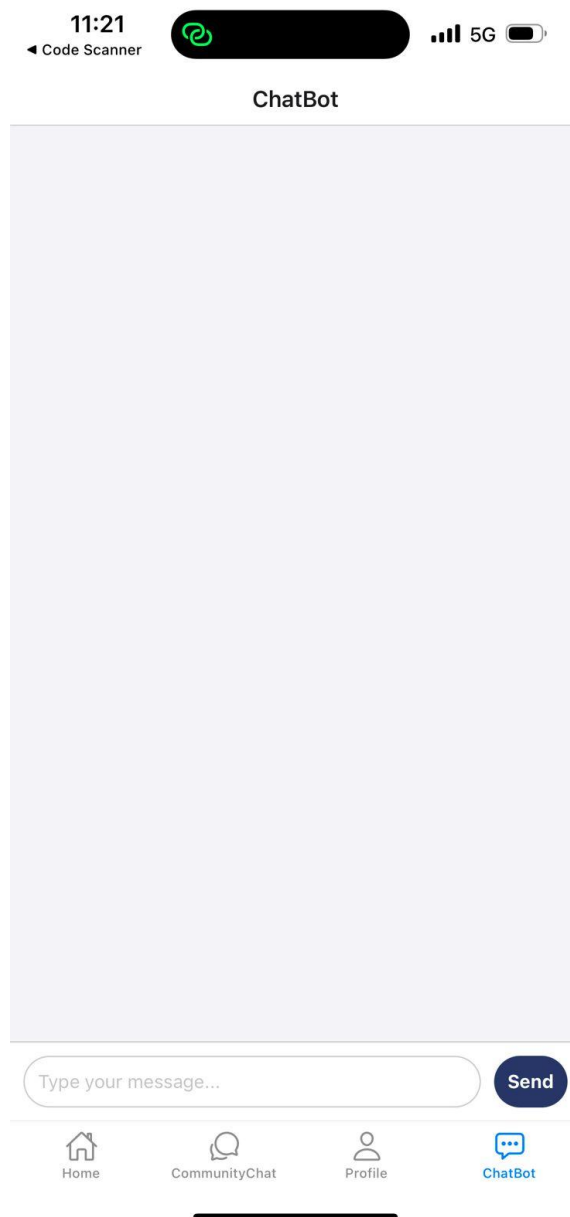


Figure B.5

AI Chatbot

The AI ChatBot screen provides instant, AI-driven assistance to juniors, offering guidance on academics, placements, and other college-related queries. It serves as a 24/7 support tool to enhance the mentorship experience

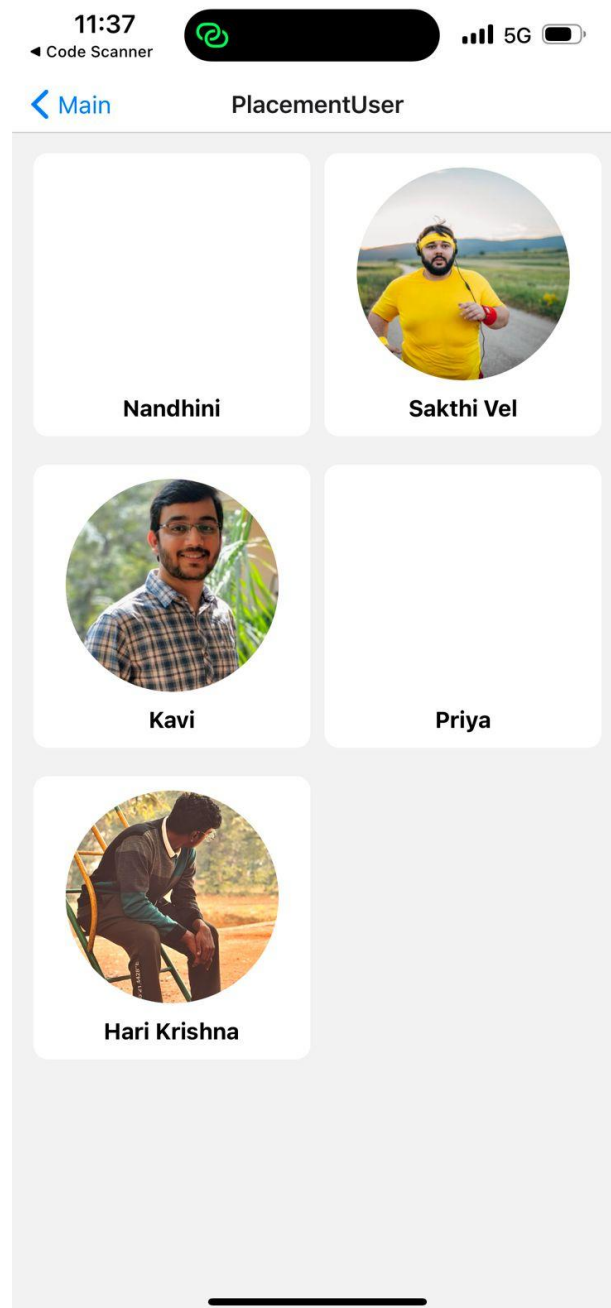


Figure B.6

Placement User Screen

The Placement Coordinator List module displays a list of seniors offering placement guidance. Juniors can easily connect with these mentors to receive support on placement strategies and opportunities.

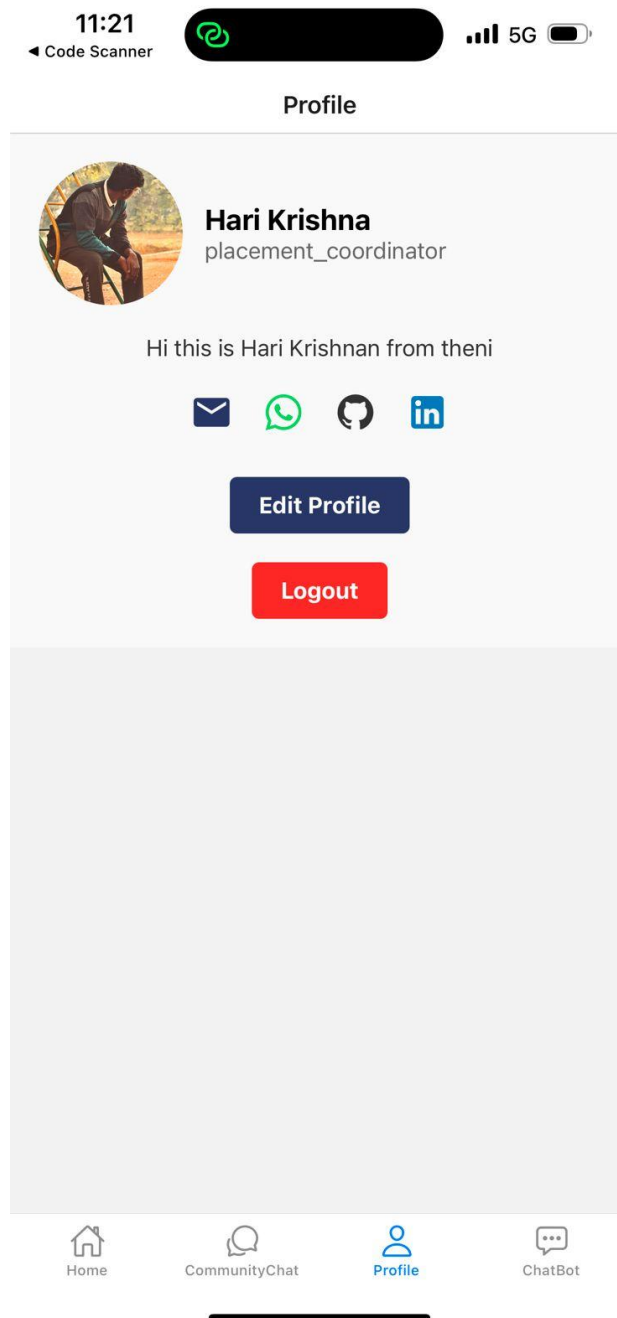



Figure B.7

Profile Screen

The Profile Screen displays the user's detailed information, including their name, department, and selected expertise or interests. It provides options to update personal details, ensuring accurate and up-to-date data for effective user engagement.

11:37
◀ Code Scanner 5G

Profile

 **Hari Krishna**
placement_coordinator

Hari Krishna

placement_coordinator

Hi this is Hari Krishnan from theni

file:///var/mobile/Containers/Data/Application/D5D2F84...

nhari.23mca@kongu.edu

9003854088

https://github.com/syssyncer

https://www.linkedin.com/in/harikrishnan-n-463a17223/

Save Cancel Logout

Home CommunityChat Profile ChatBot

Figure B.8

Profile Management Screen

The Profile Management Screen allows users to view and update their personal details, such as name, department, and areas of expertise. This ensures accurate information is maintained, enhancing user experience and effective engagement within the application.

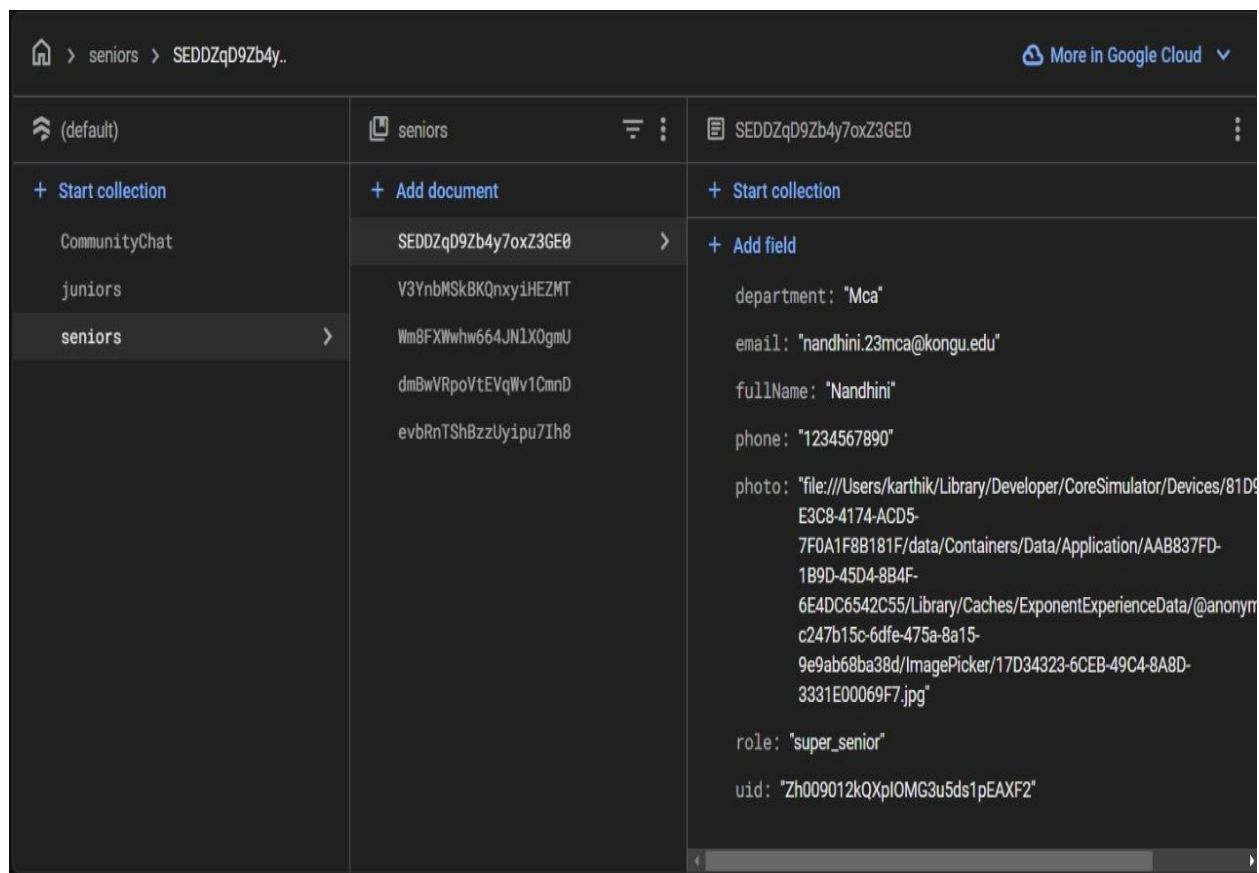


Figure B.9

Firestore database

The Figure B.9 describes every data from user side and admin side will store under the Firestore database. Information from the authentication and storage will finally store on the cloud firestore. If admin delete any information from the authentication or storage and he/she want to retrieve it, the information will store on the database so admin can retrieve easily.

REFERENCES

BOOK REFERENCE

- [1] Ethan Brown, *"React Native for Beginners: A Hands-on Guide to App Development"*, O'Reilly Media, Volume 1, 2nd Edition, 2021.
- [2] Jacob Turner, *"Mastering Firebase for App Development"*, Packt Publishing, Volume 1, 1st Edition, 2020.
- [3] Sarah Johnson, *"Full-Stack Development with React Native and Firebase"*, Manning Publications, Volume 1, 1st Edition, 2022.
- [4] David Miller, *"Practical React Native: Build Cross-Platform Mobile Applications"*, Apress, Volume 1, 1st Edition, 2020.
- [5] Emily Rogers, *"Firebase: Cloud-Driven Application Development"*, Addison-Wesley Professional, Volume 1, 2nd Edition, 2019.
- [6] Michael Carter, *"Advanced React Native and Firebase Integration"*, Wiley, Volume 1, 1st Edition, 2023.

WEBSITE REFERENCE

- [1] <https://reactnative.dev/>
- [2] <https://www.geeksforgeeks.org/react-native/>
- [3] <https://firebase.google.com/>
- [4] <https://www.w3schools.com/> — 'Learn React Native Basics'
- [5] <https://www.udemy.com/>
- [6] <https://www.youtube.com/>