الجمهورية الجزائرية الديمقراطية الشعبية
The People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جامعة محمد بوضياف بالمسيلة
University Mohamed Boudiaf of M'sila

كلية الرياضيات والإعلام الآلي
Faculty of Mathematics and Informatics

قسم الإعلام الآلي
Department of Computer Science

Domain: Mathematics and Computer Science

Thesis Presented to Fulfill the Partial Requirement
for **Master's Degree** in Computer Science

Specialty: Information Systems and Software Engineering (ISSE)

Prepared By: Omar Bourahla and Imane Bourahla

Supervised By:

Dr. Noureddine Amraoui

ENTITLED

ProgNet: A Social Network for University Community

Jury Members

Mr. Abdelbasset Barkat	President
Mr. Noureddine Amraoui	Supervisor
Mr. Abdelghani Boudaa	Examiner

Academic Year 2024/2025

Dedication

To those who planted the seeds of ambition in my heart, and nurtured them with love, prayers, and patience. To my beloved parents, the heartbeat of my journey, the source of my strength, and the reason behind my smile ,dedicate this achievement to you, for it is by Allah's grace and your unwavering support that. I have reached this milestone. To my brothers and sisters, my constant support and the warmest part of my life .To my dear friends who shared the dream and lit the way during moments of darkness .And to every teacher who taught me a word and inspired me to move forward. I dedicate this work to you all, with pride and gratitude.

B. Omar

To the one who instilled values in me and nurtured me with endless love. To the one who stayed up for my sake and sacrificed her comfort for mine. To the heartbeat of my life and the source of my strength. To my beloved **mother**, may Allah prolong her life and bless her. And to the one who taught me how to be. To the support and role model. To my dear **father**, who has always been behind every success I achieve. To my brothers and sisters. To those who stood by my side and supported me every step of the way. To my loyal friends. And to everyone who shared this journey with me and offered support, even with a kind word. I dedicate to you the fruit of my efforts and the labor of years. With deep gratitude, appreciation, and love that words cannot fully express.

B. Imane

Acknowledgement

First and foremost, we express our deepest gratitude to **Allah**, whose blessings, guidance, and strength have accompanied us throughout every step of this academic journey.

We would like to extend our sincere thanks to our supervisor, Dr. Noureddine Amraoui, for their continuous support, valuable guidance, and constructive feedback. Their expertise and encouragement were fundamental to the completion of this work.

Our heartfelt appreciation goes to the faculty members and staff of Computer Science Department at University Mohamed Boudiaf of M'Sila, whose knowledge, assistance, and academic spirit enriched our experience and contributed to our personal and professional growth.

We are especially grateful to our families, who have been our unwavering source of strength and inspiration. Their patience, love, and constant support have sustained us through every challenge and moment of doubt.

To our friends and classmates, thank you for your camaraderie, collaboration, and encouragement. The shared moments, discussions, and ideas have made this journey meaningful and memorable.

We also thank everyone who directly or indirectly contributed to the realization of this project. Every kind word, helpful suggestion, or moment of support helped shape this work into what it has become.

Finally, we dedicate this achievement to all those who believe in us and continue to inspire us to strive for excellence.

Table of Contents

List of Figures

List of Tables

General Introduction

Over the past two decades, the rapid advancement of digital communication technologies has revolutionized the way academic institutions operate. In Algeria, the Ministry of Higher Education and Scientific Research adopted a large-scale national platform called Progres, which serves as a unified system for student enrollment, course management, and administrative tracking across universities. In addition, some universities have developed their own systems to address local needs, such as the Digital Workspace (DWS) platform at the University of M'Sila. Moodle is also used as a Learning Management System (LMS) to support e-learning and organize educational content. These platforms collectively have significantly improved the management of academic and administrative activities in higher education institutions.

Problem Statement and Motivation

Despite the presence of these platforms, Algerian universities still rely heavily on external social media platforms, particularly Facebook, as the primary channel for sharing news, announcements, and academic content with students. While this approach offers convenience and widespread reach, it poses several critical challenges. Firstly, content dissemination is subject to these platforms' algorithmic filtering, meaning not all students receive timely or complete updates. Secondly, this model lacks structure: academic resources, event announcements, and student discussions become buried in an unorganized feed, making retrieval difficult. Furthermore, relying on third-party platforms raises privacy concerns and limits institutional control over data. Finally, not all students actively use or engage with these platforms, such as Facebook, resulting in fragmented communication and missed information. These limitations highlight the need for a dedicated, university-owned social network platform tailored to academic and community needs.

Proposed Solution

This work proposes ProgNet, a university-focused social networking platform tailored to the academic and social needs of students, faculty, and administrators. ProgNet stands for Progres Network, and is designed and implemented with a set of core features that directly respond to the shortcomings identified in existing solutions. These features have been successfully realized within the scope of this project and are outlined as follows:

- User Authentication and Authorization – The platform supports secure login mechanisms, ensuring that only authorized members of the university can access the system based on their roles (i.e., Student, Teacher, Head of Department, Admin).
- Profile Management – Users can edit their profile information including username, password and profile picture through a dedicated profile management interface.
- News Feed System – All users can publish posts to a centralized news feed. The Head of Department can also post department-specific announcements to targeted audiences, including academic updates and upcoming events.
- Classroom Management – Teachers can create and manage classes, post educational content, and accept student enrollment requests. Students can browse available classes, send join requests, and interact with posts shared within enrolled classes.
- Event Management – The Head of Department can create academic or social events with full details (name, time, location), and users can view and track upcoming departmental events.
- Messaging System – A private messaging system allows users to communicate securely in one-on-one conversations, supporting both formal academic interaction and informal peer communication.
- Resource and Content Sharing – Teachers can share educational resources (files, documents, and polls) within their classes. Students enrolled in those classes can access the materials.

- User Management – Heads of Departments manage user accounts within their department, while system administrators have global access to manage all users across the platform.

These integrated features provide a robust and secure environment for academic collaboration, community engagement, and institutional communication within Algerian universities.

Methodology

The development of ProgNet follows a structured software engineering approach, comprising the following stages:

- Requirements Analysis – We analyze the needs of various university stakeholders, including students, professors, and administrative staff to extract both the functional requirements and non-functional requirements that the platform must fulfil. Additionally, similar systems are studied to identify common features and limitations.
- System Design – We illustrate the overall architecture of the system. The C4 model is used to represent the different architectural layers (Context, Containers, Components), alongside UML diagrams such as sequence and activity diagrams to document user interactions with the system.
- Implementation – We develop the system as two separated applications: a backend server that handles core business logic such as authentication and class enrollment; and a mobile application as frontend for end-users.
- Deployment – We deploy the backend of the system on a dedicated VPS server, allowing full control over the runtime environment and ensuring stable and secure performance.

Report Structure

The rest of this report is structured as follows: Chapter 1 reviews existing systems to identify gaps, defines the target users and stakeholders within the university, and outlines key requirements and security considerations. Chapter 2 presents the system design using the C4 model, describing the architecture, backend, data model, and mobile app structure, supported by UML diagrams to illustrate interactions and processes. Chapter 3 discusses the system implementation and deployment, outlining the development and deployment technologies, and the presentation of the mobile application and admin dashboard. The report concludes by evaluating project outcomes, its impact on the university community, and proposing future enhancements.

Chapter 1

Business Analysis

This chapter provides a comprehensive analysis of the business context surrounding the development of the proposed system. It begins by exploring existing related systems to identify current solutions, their limitations, and opportunities for innovation. The chapter then defines the primary target users and stakeholders whose needs and expectations shape the direction of the project. A detailed requirements analysis follows, translating business objectives into functional and non-functional specifications. To ensure a user-centered design, a description .of key use cases is presented, illustrating how the system will support typical user interactions Given the increasing importance of protecting sensitive data and operations, security considerations are also discussed to address potential threats and ensure compliance with best practices. This foundational analysis sets the stage for the subsequent design and implementation phases of the system.

1.1 Related Systems

Several digital platforms have been implemented across Algerian universities to manage administrative and academic functions. At the national scale, the ministry of higher education and scientific research developed Progres, a unified system for student enrollment, course management, and administrative tracking. At the local university scale, initiatives such as the Digital Workspace (DWS) platform at the University of M'Sila have been introduced to enhance academic collaboration between students and teachers. Moodle, a powerful and flexible Learning Management System (LMS), is also widely used in various educational settings. While these systems have been widely adopted, they still do not sufficiently foster communication, collaboration, or social interaction within the university community.

Internationally, several specialized platforms have been developed to address similar needs in higher education institutions. We discuss the key features and aspects of these platforms as .summarized in Table 1.1

- ,**UniversityCube:** a comprehensive social networking platform designed for students educators, and creators. It offers a Facebook-like experience with features such as user profiles, news feeds, group collaborations, and messaging systems.

- **Portfolium:** Portfolium is an academic-oriented social networking service that allows university students and recent graduates to showcase their work through media-rich resumes.
- **CampusGroups** serves students, faculty, and staff by offering a comprehensive set of features, including user profiles, news feeds, group communications, messaging, event management, academic collaboration tools, and resource sharing. It provides real-time notifications through Firebase FCM and includes robust privacy settings and administrative moderation tools. Its strength lies in integrating academic, social, and lifestyle functionalities into a unified system.
- **MyCampus** focuses on event management, group communication, and RSVP functionalities for students, faculty, and administrators. While effective in supporting social organization within universities, it provides limited academic collaboration tools and centers mainly on event-based interactions.
- ,**Unibuddy** targets prospective and current students, emphasizing direct messaging user profiles, and event sharing. Its primary use is for student recruitment and mentorship activities, and it offers minimal support for broader academic or community collaboration.

Feature/Aspect	CampusGroups	MyCampus	Unibuddy	UniversityCube	Portfolium
Core Focus	Academic, social, and lifestyle integration	Event management and group communication	Student recruitment and mentorship	Social networking for students and educators	Academic portfolio and career showcasing
User Profiles	✓	✓	✓	✓	✓
News Feed	✓	✕	✕	✓	✕
Messaging	✓	✓	✓	✓	✕
Event Management	✓	✓	✓	✕	✕
Academic Collaboration	✓	✕(limited)	✕ (limited)	✕(limited)	✓(via project showcase)
Resource Sharing	✓	✕	✕	✓	✓
Real-time Notifications	✓ (via Firebase FCM)	✕	✕	✕	✕
Privacy Settings	✓(robust)	✕	✕	✕	✓(moderate)
Admin Moderation Tools	✓(robust)	✕	✕	✕	✕

Table 1.1: Summary and Comparison of Related Systems

Despite the notable successes of these international systems, they often fall short when applied to the Algerian university context. They either lack cultural adaptation to local academic environments or fail to meet the specific social and institutional needs of Algerian students and staff. Furthermore, the widespread informal use of general-purpose social networks like Facebook, Telegram, and Discord introduces concerns regarding data privacy, institutional control, and fragmented user experiences that do not align with the goals of higher education institutions.

In response to these challenges, we propose ProgNet, a dedicated, secure, and culturally adapted social networking platform designed specifically for Algerian universities. ProgNet stands for Progres Network, and is named after Progres, the integrated platform of the Algerian university. ProgNet aims to bridge the communication and collaboration gaps left by existing administrative platforms, offering a structured, safe, and community-centered environment that promotes academic engagement and enriches campus life.

1.2 Target Users and Stakeholders

ProgNet is designed to serve a broad range of users within the university community. Key stakeholders include:

- ,**Students:** Represent the most active user group, engaging in academic discussions social interactions, and university events.
- ,**Teachers:** Interact with students by sharing academic content, posting announcements and participating in discussions.
- **Head of Department (HoD):** Manages departmental user accounts, publishes department-specific announcements, and ensures effective academic communication within the department.
- **System Administrator:** Oversees all user accounts, enforces general platform policies, and ensures the security and continuity of system operations.

This diversity of users highlights the importance of designing a comprehensive system that responds to the specific needs and responsibilities of each stakeholder, thus ensuring a consistent and effective user experience across all levels of interaction.

1.3 Requirements Analysis

The development of ProgNet is grounded in a detailed requirements analysis phase, which takes into account the needs of the university community.

1.3.1 Functional Requirements

User Stories is a widely recommended way to describe functional requirements, especially in Agile and user-centered development methodologies. In the following, we describe The functional requirements of ProgNet as user stories categorized by functionality :

User Authentication

- As a university member, I want to log in using a unique username and password so that I can securely access the platform and ensure it's a trusted environment.
- As a system administrator, I want to create accounts for users' with university affiliation so that only eligible members gain access to ProgNet.

Profile Management

- As a user, I want to manage and update my personal profile information, including my username, password, and bio, so that I can maintain an accurate and secure representation of my identity within the university community.
- As a user, I want to be able to change my profile picture so that I can personalize my presence and enhance recognition among peers.

News Feed

- As a teacher, student, or Head of Department, I want to create and publish posts on a centralized news feed so that I can share relevant updates and academic content with the university community.
- As a Head of Department, I want to post official announcements on the department's dedicated feed so that department members stay informed.
- As a student or teacher, I want to access both general and departmental news feeds so that I don't miss important information or activities.

Messaging System

- As a user, I want to send private messages to other users so that I can communicate directly with peers and faculty.

Event Management

- As a Head of Department , I want to create and manage events so that I can organize academic or social activities on campus.
- As a student or teacher, I want to view and join events and receive reminders so that I can stay informed and participate in upcoming activities.

Resource Sharing

- As a teacher, I want to upload and share educational resources and files within the classroom page so that students can access relevant academic materials.
- As a student, I want to access shared resources in the classrooms I am enrolled in so that I can benefit from the materials provided by my instructors.

1.3.2 Non-Functional Requirements

The Non-functional requirements of ProgNet include:

- **Scalability:** ProgNet must be designed with scalability in mind to accommodate future growth. As more universities join the platform and the user base expands, the system architecture should allow seamless scaling without degradation in performance, ensuring continued responsiveness and reliability.
- **Usability:** ProgNet frontend should provide a user-friendly and intuitive interface, making it accessible to all university community members regardless of their level of technical expertise. Attention should be given to clear navigation, responsive design, and minimizing the learning curve for new users.
- **Reliability:** High system availability is crucial for maintaining user trust. ProgNet must ensure 24/7 operational uptime with minimal service interruptions. It should incorporate robust error-handling mechanisms, automatic recovery procedures, and regular system maintenance to minimize downtime and ensure continuous service delivery.

1.4 Use Cases

In the following table (Table 1.2), we map the defined user stories into use cases organized by major functional areas. Each use case summarizes the related user goals and interactions described in the user stories.

Use Case	**Actors**	**Description**
Authenticate User	,Student, Teacher Head of Department	Users log in using their institutional credentials
Manage User Profile	,Student, Teacher Head of Department	,Users can update their institutional credentials, password and profile picture
Create and Manage Posts	Student, Teacher and Head of Department	Users can create and manage general posts to share updates or academic content
Department Posts	Head of Department	The Head of Department publishes department-specific posts in the Administ section
Create and Manage Classes	Teacher	Teachers can create class pages, manage class content, and upload related files
Student Enrollment	Student	Students can enroll in classes created by teachers to access class materials
Create and Manage Events	Head of Department	The Head of Department posts department-specific events with relevant details
Messages	All Users	Users communicate privately through a secure one-on-one messaging system
Manage Department Accounts	Head of Department	The HoD manages user accounts related to their department
Manage All Accounts	System Administrator	The administrator manages all user accounts and enforces platform policies

Table 1.2: Use Cases Description

1.5 Security Consideration

Given the nature of ProgNet as a social network for the university community, ensuring security and data privacy is of paramount importance. The system handles various types of sensitive information, including personal user data, academic communications, classroom interactions, and administrative processes. Accordingly, several security measures should be considered during development and deployment:

- User Authentication and Authorization - The system should employ a secure authentication mechanism to verify user identities in a reliable and stateless manner. Upon successful login, the system should authorize access to protected resources without the need for repeated authentication. Access should be strictly restricted based on clearly defined user roles (student, teacher, HoD, administrator), effectively preventing unauthorized access to sensitive data and functionalities.

- Data Protection and Privacy– To protect users' personal and academic information, the system should adhere to best practices in data storage. Critical operations such as messaging and classroom management should be secured against tampering or unauthorized manipulation.

- User Settings and Consent– Users should be provided with settings to control preferences such as language selection, profile updates, and display modes (e.g., dark mode or following the system theme).

- Secure Messaging and Communication– The messaging system should be designed to ensure confidentiality between communicating users. The exchanged data must be protected and inaccessible to unauthorized parties, fostering a secure academic communication environment.

1.6 Conclusion

This chapter outlined the foundation of the ProgNet system by analyzing existing systems, defining target users and requirements, and highlighting key security measures. This analysis sets the stage for the subsequent design and implementation phases of the system.

Chapter 2

System design

This chapter outlines the architectural and technical design of the ProgNet system. ProgNet is designed to support the functional and non-functional requirements identified in the previous chapter. This chapter describes the main components of the system, how they interact, and the rationale behind design decisions to ensure a scalable, maintainable, and secure system.

2.1 System Architecture

ProgNet adopts a modular, layered client-server architecture, where the mobile application communicates securely with the backend server via APIs. To illustrate the system's architecture we use the C4 model to present two perspectives: the Context view, showing interactions with external actors, and the Container view, detailing the main software components and their responsibilities.

2.1.1 Context

Figure 2.1 illustrates the system context of ProgNet outlining the primary actors and their interactions. The core actors include Students, who can access their university information and enroll in classes; Teachers, who are university professors able to create classes and publish content to students; and the Head of Department, responsible for verifying and managing both student and teacher accounts. An Admin also exists in the system, tasked with overseeing and managing accounts belonging to department heads. ProgNet integrates with third-party ,services such as SupaBase, used for storing and retrieving images and videos, and Firebase which facilitates the sending of notifications to users.

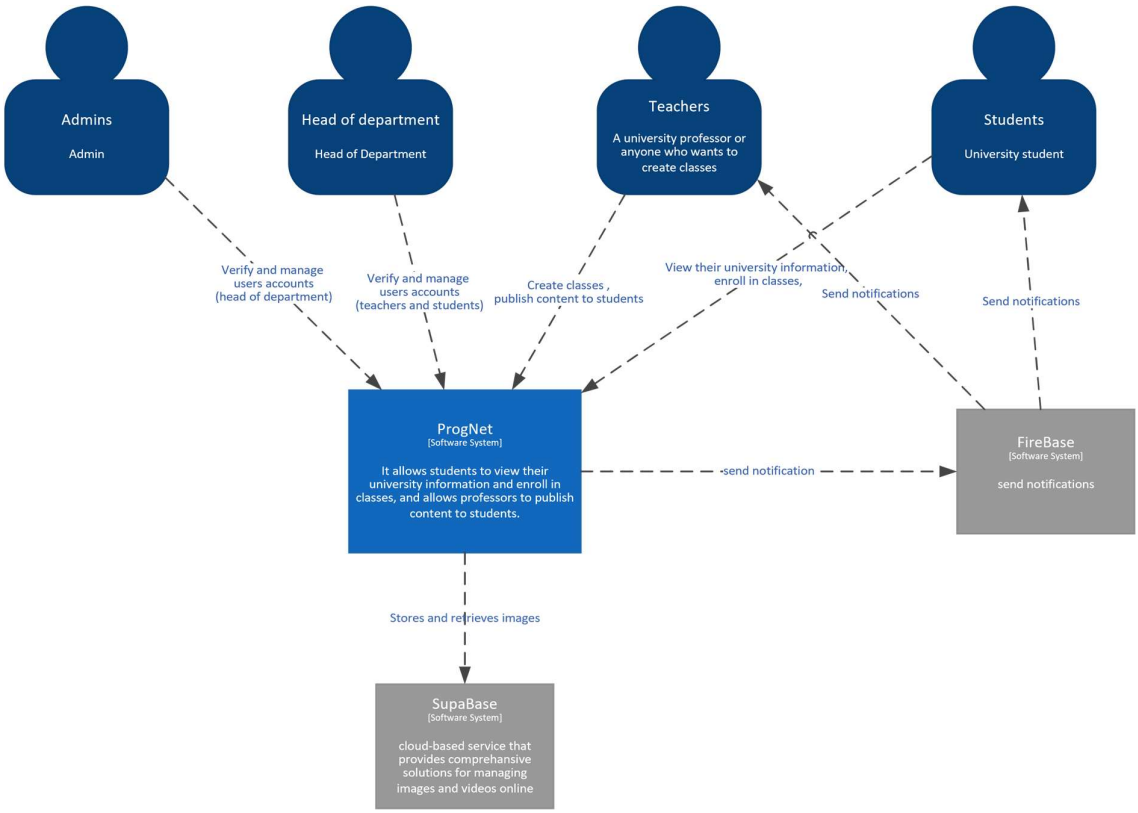

Figure 2.1: System Context

2.1.2 Containers

Figure 2.2 illustrates the containerized architecture of ProgNet, showcasing its main components and their interactions. The system features multiple user interfaces, including a ,Mobile UI and Web UI (Interact with the admin dashboard features) built using Flutter enabling users to interact with the application's functionalities. The Backend API, developed with Node.js, handles core business logic and essential operations such as authentication, class enrollment, content publishing, and notifications. The UI components communicate with this server via JSON/HTTPS requests. Data storage is managed through a MySQL database containing user registration details, encrypted credentials, and system logs. Additionally, the system integrates with SupaBase for media file management and Firebase for real-time notifications, enhancing interactive content handling capabilities.

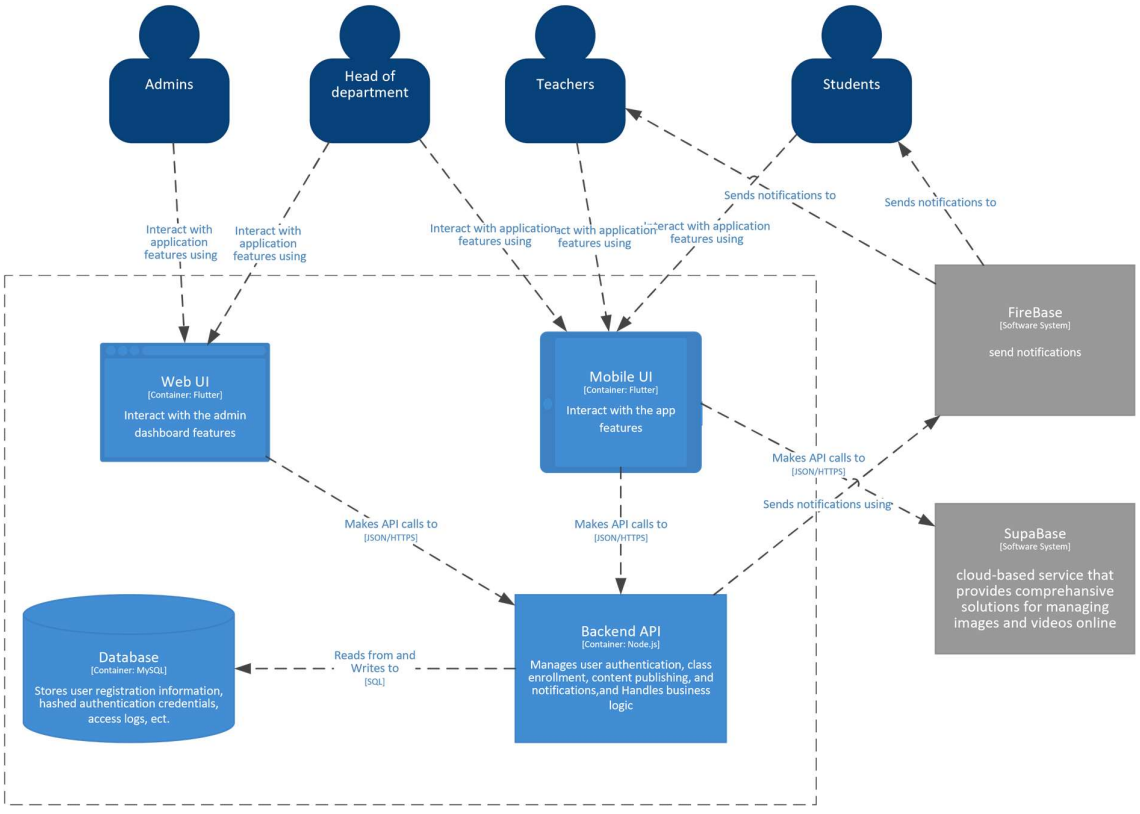

Figure 2.2: System Containers

2.2 Back-end Server

Figure 2.3 illustrates the components of the backend server system. The server consists of several core modules that collaborate to process requests and manage data. The system .includes a Router that receives HTTP requests and routes them to the appropriate controllers ,The Middleware handles essential preprocessing tasks such as request validation authentication, and authorization. The Controller contains the business logic for processing user-related requests like creating posts or adding students, while also managing notification)delivery through Firebase. For database operations, the Object-Relational Mapping)ORM tool executes database queries and retrieves results from the MySQL database, which stores user registration details, encrypted authentication credentials, access logs, and other critical data. This integrated structure ensures efficient and secure processing of data and requests throughout the system.

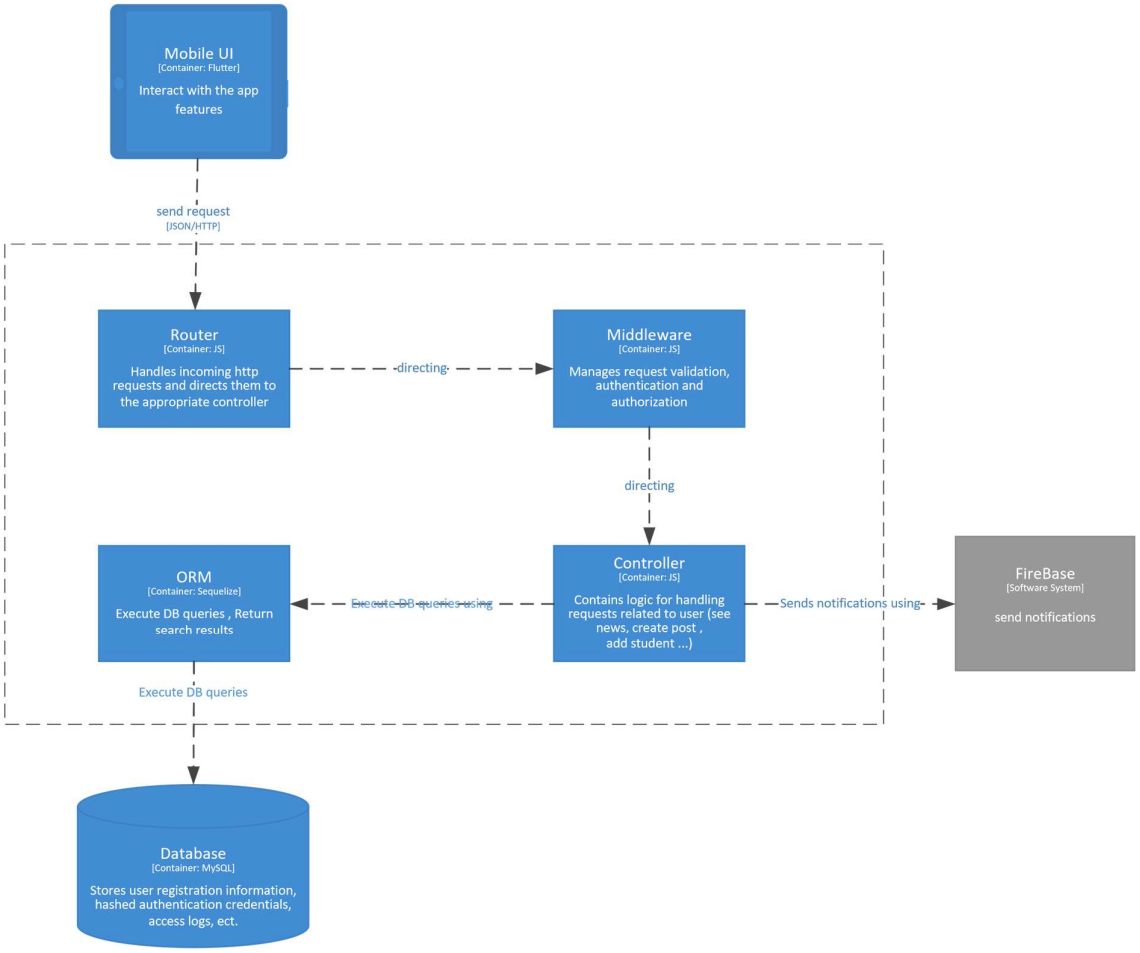

Figure 2.3: Backend Server Components

2.2.1 Data Model

Figure 2.4 presents the Entity-Relationship Diagram (ERD) that outlines the fundamental structure of the system's database. This model includes core entities such as the User, which contains basic information including name, email, and hashed password, along with the Teacher and Student entities that are linked to the User through a shared identifier. The diagram also features additional entities like News (containing content and files), Events (storing location and time details), and Classes (associated with teachers and students). The relationships between these entities are clearly depicted, showing connections such as users' affiliations with Departments and Faculties, as well as the links between Posts, Comments, and their respective classes and users. This model provides a comprehensive overview of the data structure and their interrelationships, facilitating understanding of how information is stored and processed within the system.

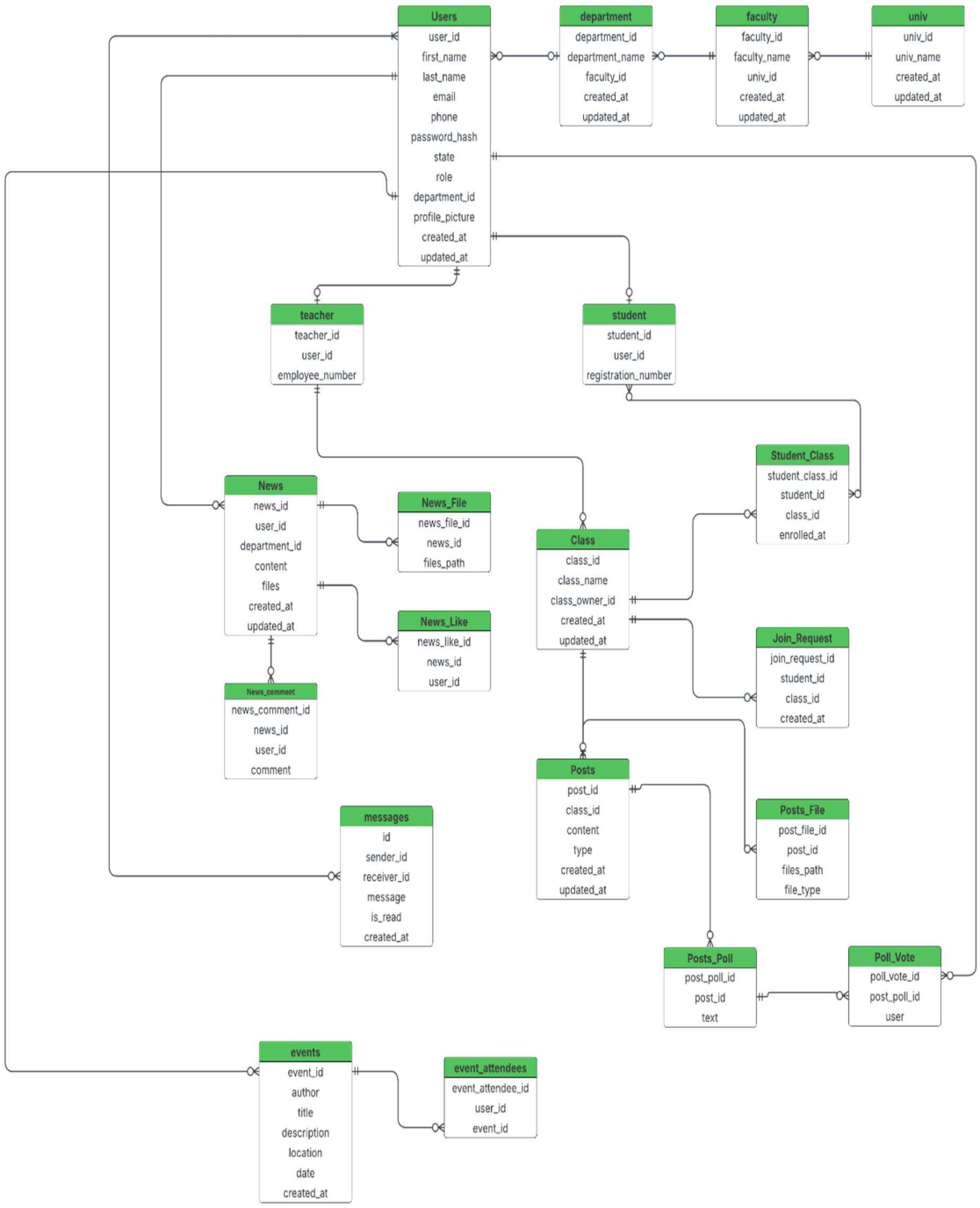

Figure 2.4: Data Model

In our data model, we have the following entities, where each entity is designed to reflect a specific function within the system :

User – Stores information for all users**(**students, teachers, HoDs, admins**)**

- user_id**–** Unique identifier for the user.
- first_name**–** User's first name.
- last_name**–** User's last name.
- email**–** User's email address.
- phone**–** Contact phone number.
- password_hash**–** Encrypted password for secure login.
- state: Account status (e.g., active, inactive.
- role: Role type (student, teacher, HoD, admin.
- department_id**–** Links the user to a department.
- profile_picture**–** Profile image path or URL.
- created_at**–** Timestamp of account creation.
- updated_at**–** Timestamp of last update.

Department – Defines academic departments within faculties

- department_id– Unique department identifier
- department_name– Name of the department
- faculty_id–ID of the faculty to which it belongs
- created_at / updated_at– Record tracking timestamps

Faculty – Stores information about faculties

- faculty_id**–** Unique faculty identifier
- faculty_name**–** Name of the faculty
- univ_id**–** Linked university ID
- created_at / updated_at**–** Record timestamps

Univ(University) – Contains basic university data

- univ_id**–** Unique university identifier
- univ_name**–** Name of the university
- created_at / updated_at**–** Timestamps for tracking changes

Class – Represents class entities created by teachers

- class_id– Unique class identifier
- class_name– Name/title of the class
- class_owner_id– User ID of the teacher who owns the class
- created_at / updated_at– Timestamps

Student – Links student to user account

- student_id– Unique identifier for the student
- user_id– Linked user account
- registration_number– Student's official registration number

Teacher – Links teacher to user account

- teacher_id– Unique identifier for the teacher
- user_id– Linked user account
- employee_number– Teacher's employment number

Student_Class – Associates students with enrolled classes

- student_class_id– Unique ID for this relationship
- student_id– The student enrolled
- class_id– The class joined
- enrolled_at– Enrollment timestamp

News – Holds department or general news posts

- news_id– Unique identifier for the post
- user_id– Author of the news
- department_id– Related department
- content– Textual content of the post
- files– Associated media or documents
- created_at / updated_at– Timestamps

News_File – Stores files linked to news

- news_file_id– Unique ID
- news_id– Linked news post
- files_path– File location

News_Like – Captures user likes on news posts

- news_like_id– Unique ID
- news_id– News post liked
- user_id– User who liked it

News_Comment – Stores user comments on news posts

- news_comment_id– Unique ID
- news_id– Related post
- user_id– Comment author
- comment– Text of the comment

Posts – Posts shared within classes

- post_id– Unique ID of the post
- class_id– Class to which it belongs
- content– Body of the post
- type– Type of post(e.g., text, poll)
- created_at / updated_at– Timestamps

Posts_File – Files attached to class posts

- post_file_id– Unique file record
- post_id– Related post
- files_path– File location
- file_type– Type of file(PDF, image, etc)

Posts_Poll – Poll options in posts

- post_poll_id– Poll option ID
- post_id– Associated post

- text– Option text

Poll_Vote – Votes on poll options

- poll_vote_id– Unique vote ID
- post_poll_id– Selected option
- user_id– Voter ID

Events – Represents academic or social events

- event_id– Event identifier
- title – Name of the event
- description– Description of the event
- location– Where it will be held
- Date – Schedule of the event
- Author – Creator of the event
- created_at – Timestamps

Event_Attendees – Users attending an event

- event_attendee_id– Unique ID
- user_id Attendee
- event_id Event attended

Messages – Messages sent in chat

- message_id– Message identifier
- message_room_id Conversation context
- sender_id / receiver_id– Communication parties
- content– Message body
- sent_at / read_at– Delivery timestamps

Join_Request – Student requests to join classes

- join_request_id– Request ID
- student_id– Requesting student
- class_id– Target class
- created_at– Request time

2.2.2 Router

The back-end server is exposed to the mobile application through a set of public and protected API endpoints. The router module is responsible for handling HTTP requests coming to different API endpoints to determine which controllers and middleware should respond to various actions and inputs from clients. In the following, we present different API endpoints categorized by users, student, teacher, HOD, and admin endpoints.

Users Endpoints

The following endpoints are available for any logged-in user of the application:

- Users Routes : */users*

Method	Endpoint	Result
POST	/login	Login to account
POST	/logout	Logout from account
POST	/edite	Edit user data
GET	/userData	Get all user data

Table 2.1: User – Authentication and Profile Management Routes

- News Feed Routes: */news*

Method	Endpoint	Result
POST	/create	Create a post
GET	/seeAll	See all posts in the department
GET	/seeAllByFaculty	See all posts in the faculty
GET	/seeAllByUniv	See all posts in the university
GET	/seeAllByUser	See all posts created by a user
GET	/seeAllAdministrative	See all posts in the administrative
POST	/like	Give the post like
DELETE	/unLike	Remove the like from a post
POST	/addComment	Add a comment in a post
GET	/seeComment	Get the comments in a post
POST	/edite	Edit a post
DELETE	/deleteNews	Delete a post

Table 2.2: News Feed – Creation, Interaction, and Retrieval Routes

Student Endpoints

The following endpoints are only available for authenticated students:

- Class Routes: */class*

Method	Endpoint	Result
GET	/getEnrolledClasses	Get list of enrolled classes
POST	/getAvailableClasses	Get list of available classes
POST	/sendJoinReq	Send Join Request to class
DELETE	/deleteJoinReq	Delete Join Request to class
POST	/voteInPoll	Vote in poll post
GET	/getPostByClassId	Get posts in the class

Table 2.3:Student – Class Enrollment and Interaction Routes

Teacher Endpoints

The following endpoints are only available for authenticated teachers:

- Class Routes: */class*

Method	Endpoint	Result
POST	/create	Create a class
POST	/accepteJoinReq	Accept join Request to class
DELETE	/deleteJoinReq	Delete Join Request to class
GET	/getTeacherClasses	Get the classes of Teacher
GET	/getClassStudents	Get the students in the class
GET	/getClassJoinRequests	Get the join Request for a class
DELETE	/deleteStudent	Delete student from class
POST	/createPost	Create a post in the class
GET	/getPostByClassId	Get posts in the class
POST	/createPostPoll	Create a post of type poll in the class
DELETE	/deletePost	Delete post from class
DELETE	/deleteClass	Delete class
POST	/changeClassName	Change the class name

Table 2.4: Teacher – Class and Content Management Routes

HoD Endpoints

The following endpoints are only available for authenticated users with the role of Head of Department(HoD):

- Users Routes : */users*

Method	Endpoint	Result
POST	/crate	Create user account
DELETE	/delete	Delete user account
POST	/editeFromAdmin	Edit user data
GET	/seeAllByDepartmentId	See all users in the department
GET	/userDataById	Get all user data

Table 2.5: Head of Department – User Management Routes

- News Feed Routes: /news

Method	Endpoint	Result
POST	/ createAdministrative	Create administrative post
GET	/ seeAllAdministrative	See all posts in the administrative

Table 2.6: Head of Department – News Feed Routes

System Administrator Endpoints

The following endpoints are only available for authenticated admin users:

- Users Routes : */users*

Method	Endpoint	Result
POST	/create	Create user account
DELETE	/delete	Delete user account
POST	/editeFromAdmin	Edit user data
GET	/seeAllByDepartmentId	See all users in the department
GET	/userDataById	Get all user data
GET	/seeAll	See all users

Table 2.7: Head of Department – News Feed Routes

2.2.3 Middleware

Middleware functions in the backend server of system act as intermediaries between incoming HTTP requests and the backend logic. They perform essential pre-processing tasks to ensure requests are valid, authenticated, and authorized before reaching the controller. The middleware functions include:

- token: Verifies the authenticity and validity of the JWT token included in the HTTP request headers And Decodes the JWT token to extract user-specific information, such as the user ID and role, for further processing.
- dataValidator: Ensures that all required data fields are present and correctly formatted in the request payload (e.g., during user profile updates or class creation).
- roleChecker: Validates the user's role (e.g., Student, Teacher, HOD, or Admin) to enforce role-based access control (RBAC) for protected endpoints.

2.2.4 Controller

The controller module is responsible for executing the core business logic of the ProgNet system. Once a request passes through the middleware, the router directs it to the appropriate controller, which processes the request, interacts with the ORM client for database operations, and returns an HTTP response. For example, when a student submits a join request for a class (POST /class/sendJoinReq/), the controller validates the request, updates the database via the ORM client, and returns a success/failure response.

The key responsibilities of the Controller include:

- User Management: Handling authentication (login/logout), profile updates, and role-specific actions (e.g., student enrollment or teacher class creation).
- Academic Operations: Managing class enrollments, processing join requests, and facilitating content publishing (e.g., posts, polls, or news).
- Communication: Handling message exchanges between users and event management (e.g., creating events or tracking attendees).
- Notifications: Triggering Firebase notifications for events like class updates or new messages.
- Data Retrieval: Fetching user data, class lists, or department-specific information based on request parameters.

2.2.5 ORM Client

The ORM (Object-Relational Mapping) client abstracts database interactions by mapping MySQL tables to JavaScript objects, simplifying CRUD operations. It is important to note that this system utilizes a third-party ORM client, namely **Sequelize**, to facilitate these mappings and operations. For instance, when a teacher creates a poll post (POST /class/createPostPoll/), the ORM client inserts records into the **Posts** and **Posts_poll** tables while maintaining referential integrity.

The key features of the ORM Client include:

- **Database Abstraction**: Translates JavaScript method calls into SQL queries (e.g., findAll() to SELECT * FROM users).
- **Relationship Management**: Handles associations between entities (e.g., User ←→ Student or Class ←→ Posts) using foreign keys.
- **Query Optimization**: Supports complex queries (e.g., fetching enrolled classes with student counts) while minimizing manual SQL.
- **Transaction Support**: Ensures data integrity during multi-step operations (e.g., deleting a class and its associated posts).

2.3 Mobile Application

This section presents the front-end architecture of the system, where the components are organized in a structured manner to ensure seamless interaction between the user and the application, with each part performing its specific role efficiently.

2.3.1 Components

Figure 2.5 outlines the front-end architecture consisting of interconnected components. The Routing module directs requests to appropriate views, while the API component manages communication with the Node.js Backend API that handles authentication, enrollment, content management and notifications. The View layer displays interactive screens that send user inputs to the Controller for processing actions like news viewing or post creation. The system integrates SupaBase for cloud-based media storage and management, ensuring efficient handling of images and videos. This structure maintains clear separation between presentation (Views), logic (Controller/API) and data (Backend/SupaBase), enabling smooth user interactions while delegating specialized tasks to appropriate services.

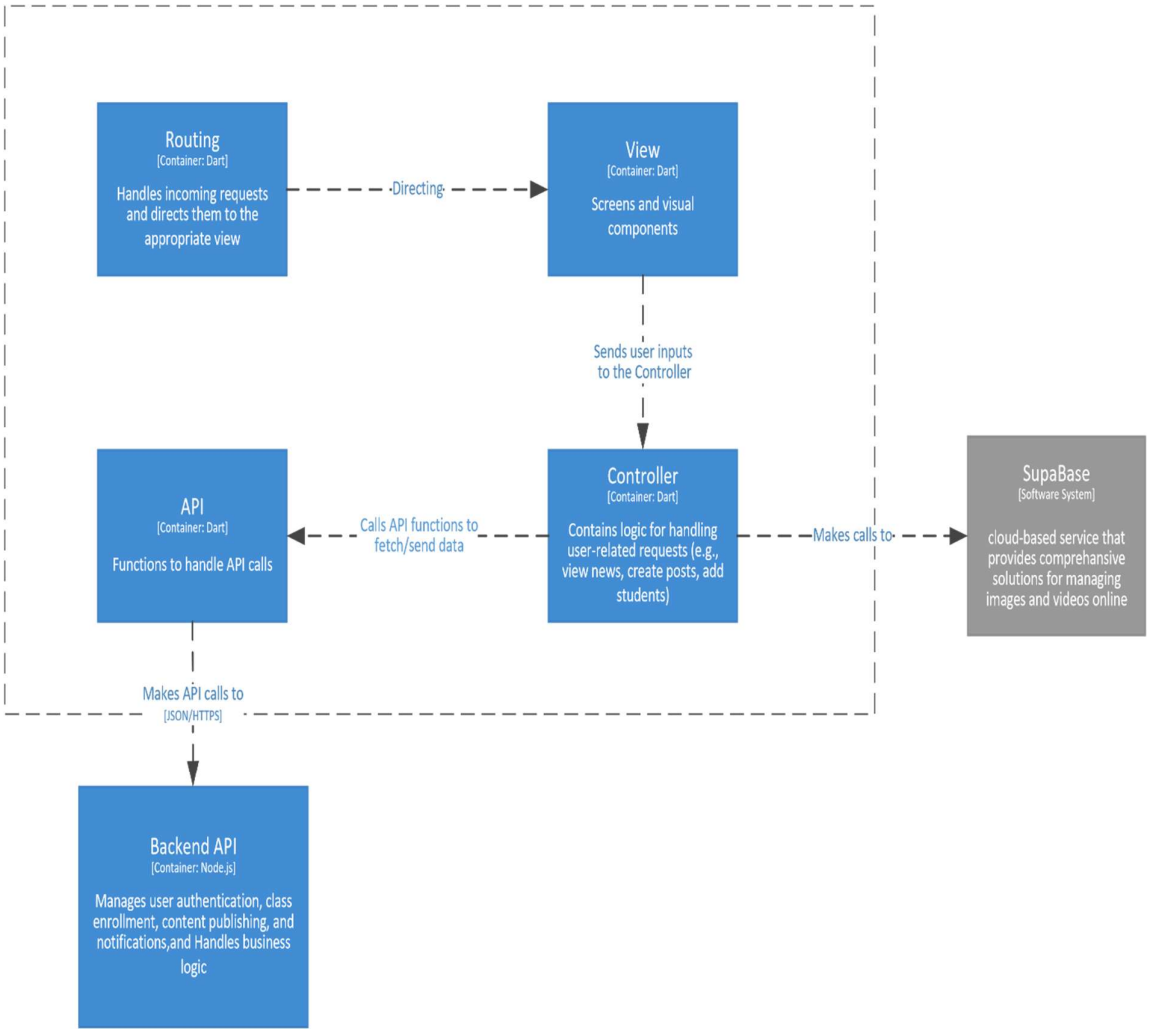

Figure 2.5: the front-end architecture

2.3.2 Activity Diagrams

In this section, we use activity diagrams to represent the sequence of activities and steps that a user performs within the system, such as logging in, creating posts, joining a class, or sending messages. These diagrams help in understanding the workflow logic of each functionality, showing how the user interacts with the system and how the application processes each step. This contributes to a precise and efficient design and implementation of both the user interfaces and backend services.

The user Authentication Process (Login and Logout): Illustrates user login and logout, including credential validation and session handling

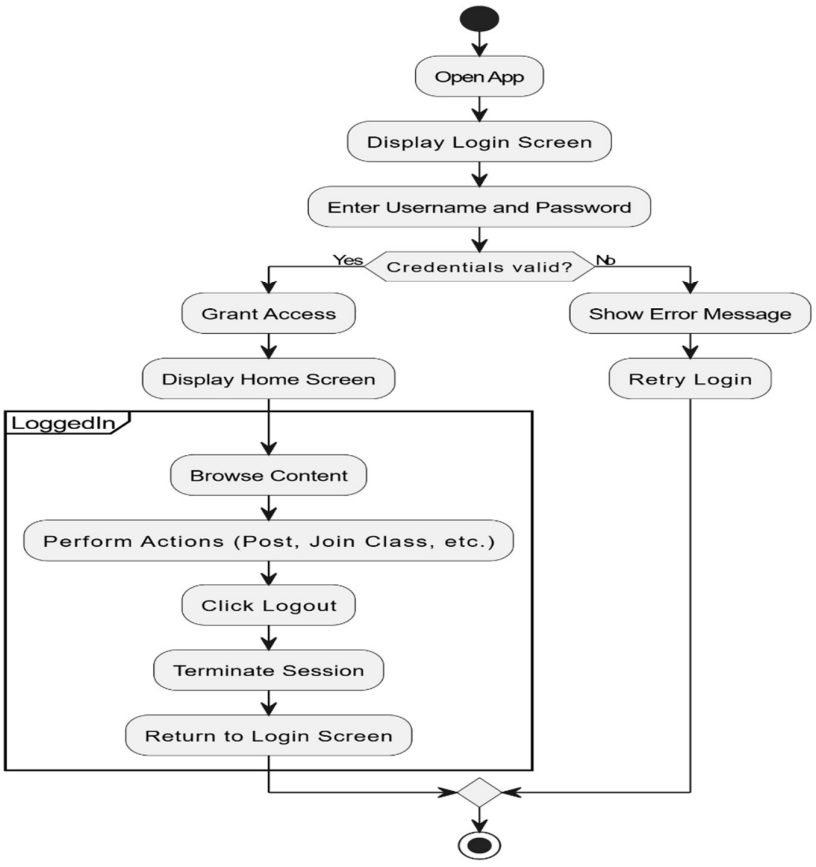

Figure 2.6: User Authentication Process (Login and Logout)

Users create, edit, or delete post : Shows how users create, edit, or delete posts through backend processing

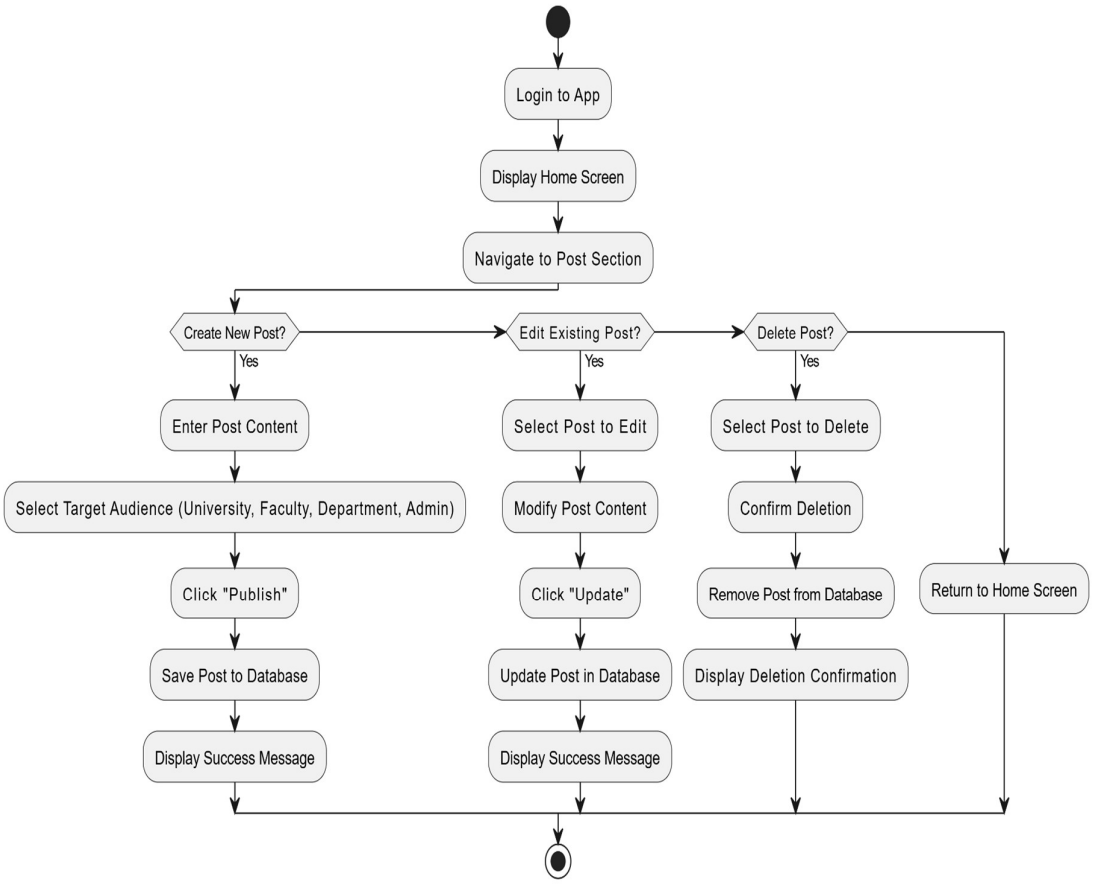

Figure 2.7: Exchange Messages Between Users

Teacher Creates Classes and Managing : Depicts how a teacher manages class records (create, edit, delete)

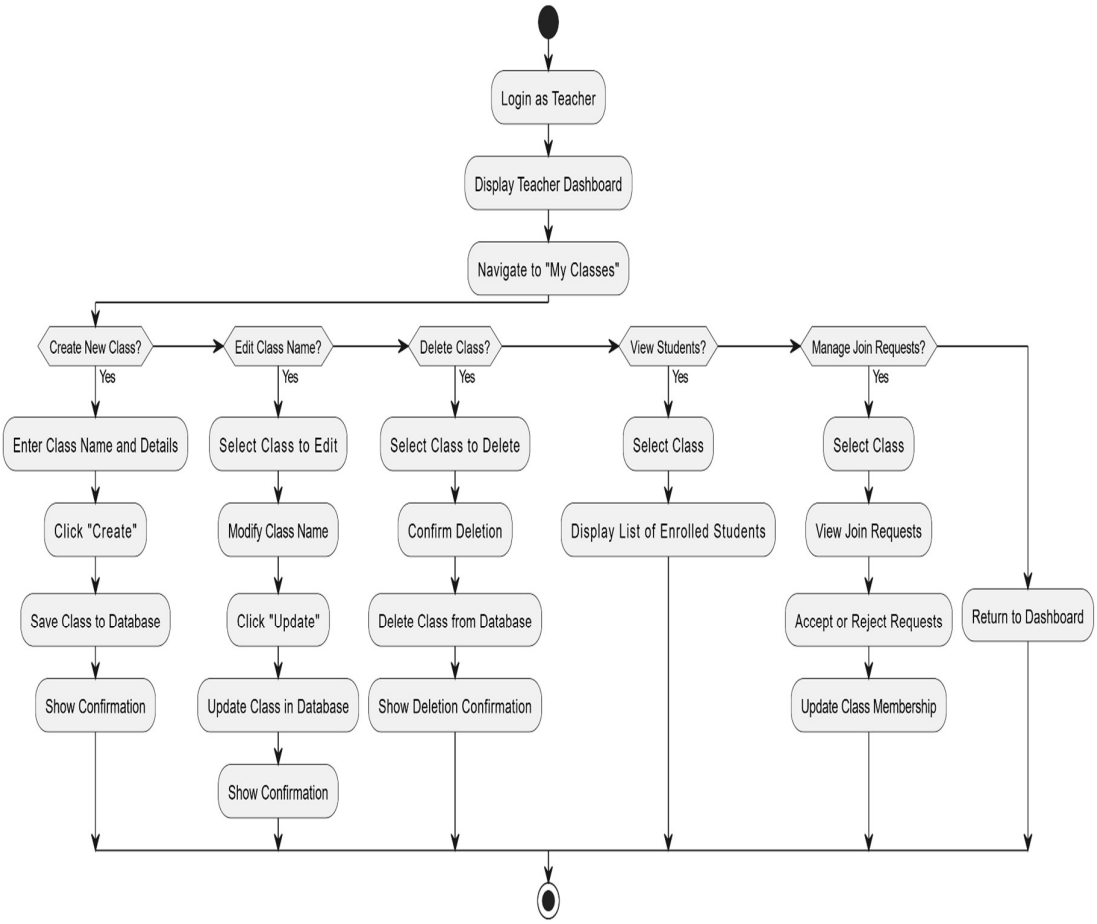

Figure 2.8: Teacher Class Management

Teacher Creates Class Posts : Explains teacher post creation in class and automatic student notifications

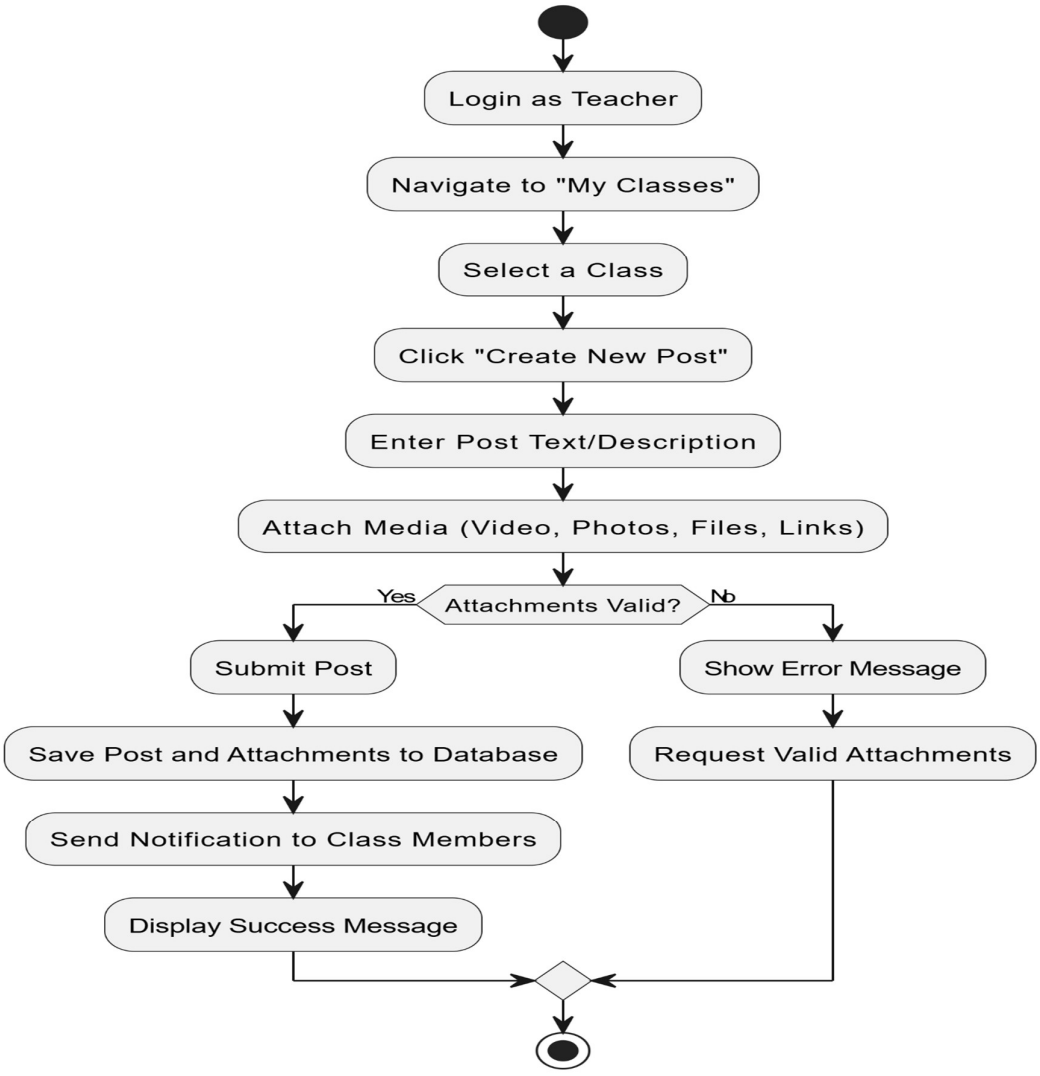

Figure 2.9: Teacher Posts in Class

Student Enrollment in a Class : Describes how a student requests to join a class and awaits teacher approval

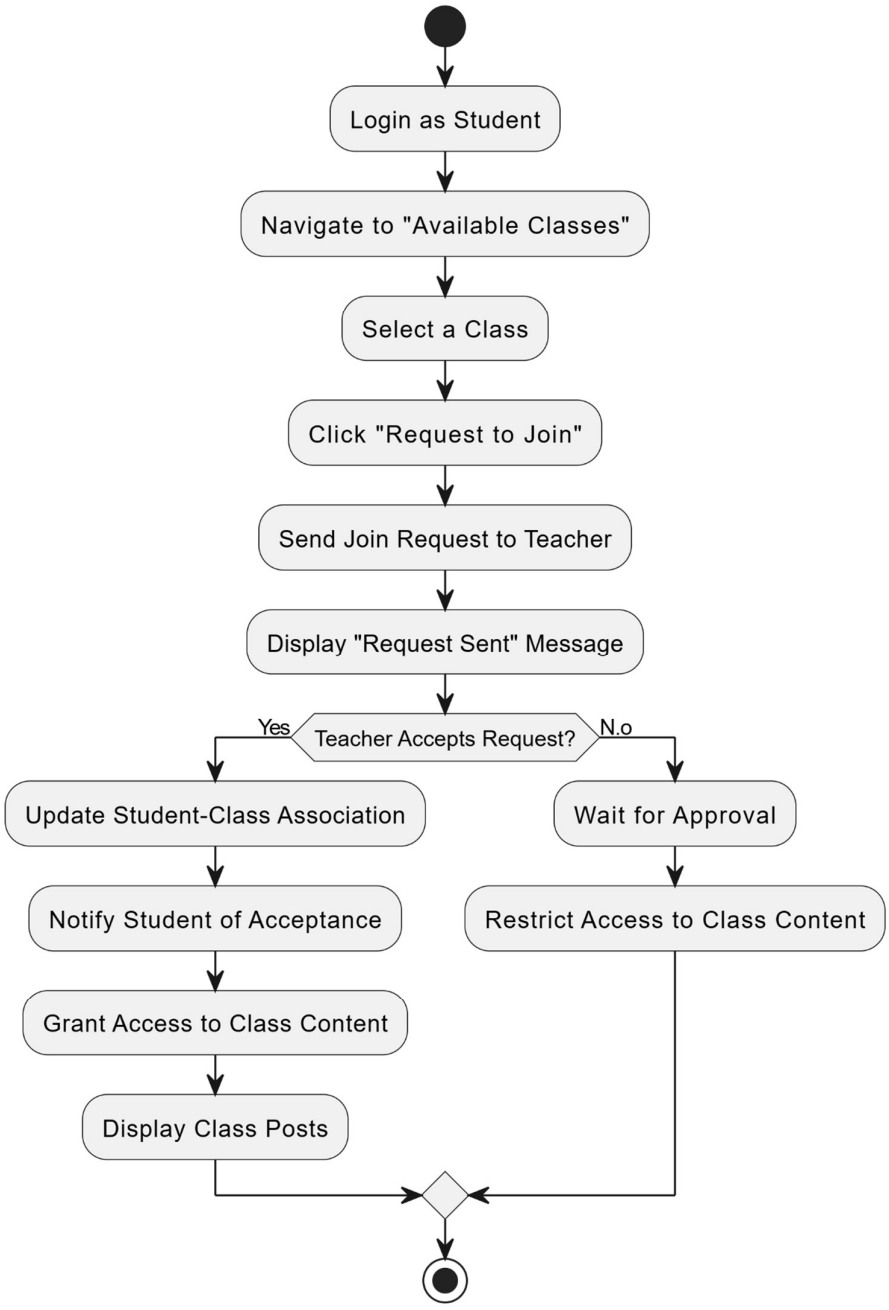

Figure 2.10: Student Enrolls in a Class

HoD Managing Event Posts : Presents how the Ho D manages event posts and notifies users of new events

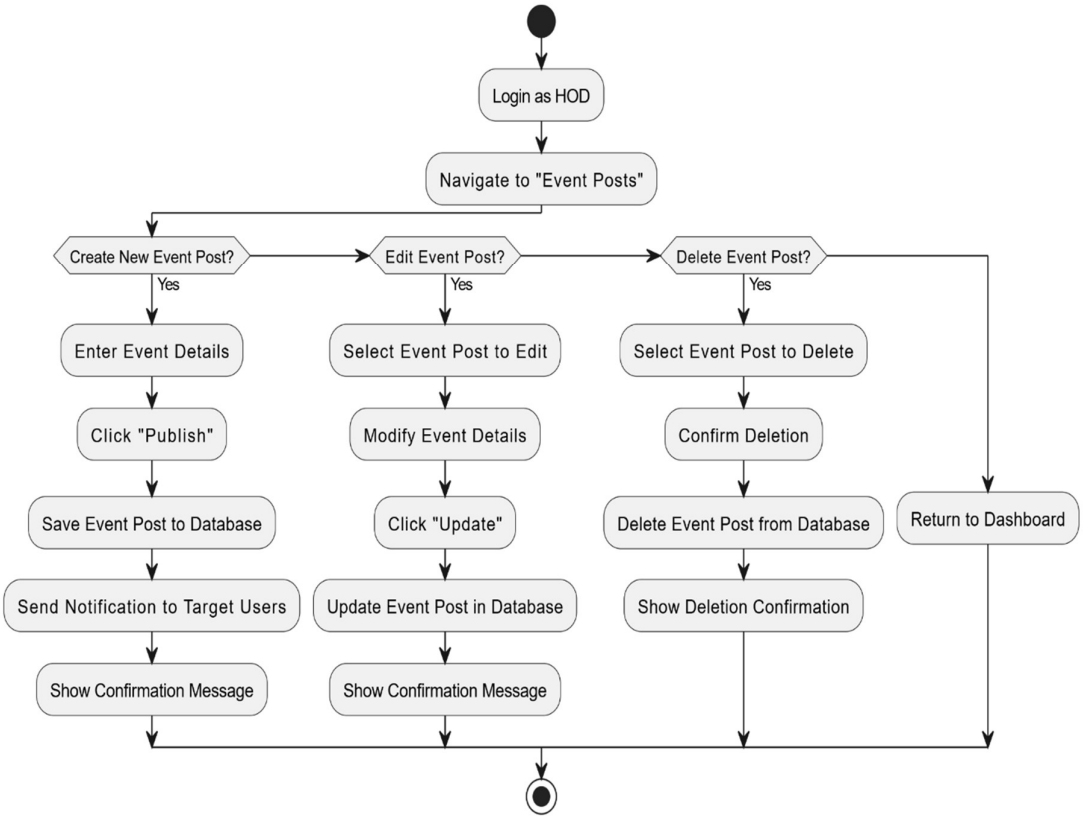

Figure 2.11: HoD Event Post Management

Message Exchange Between Users : Outlines the message exchange between users from submission to delivery

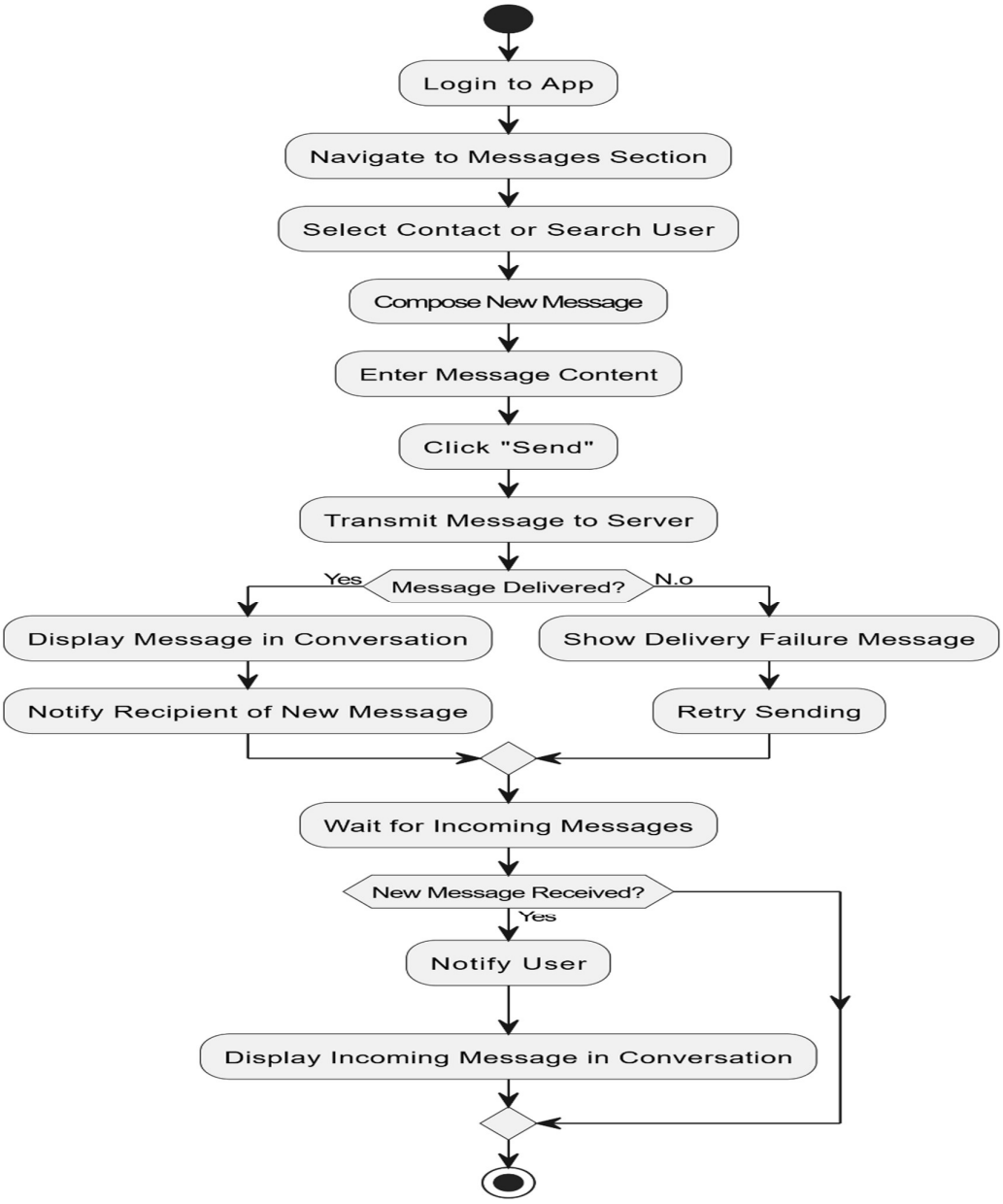

Figure 2.12: Messaging Between Users

HoD Manages Department-Specific: User Accounts DemonstratesHoD’ s ability to manage user accounts within a department

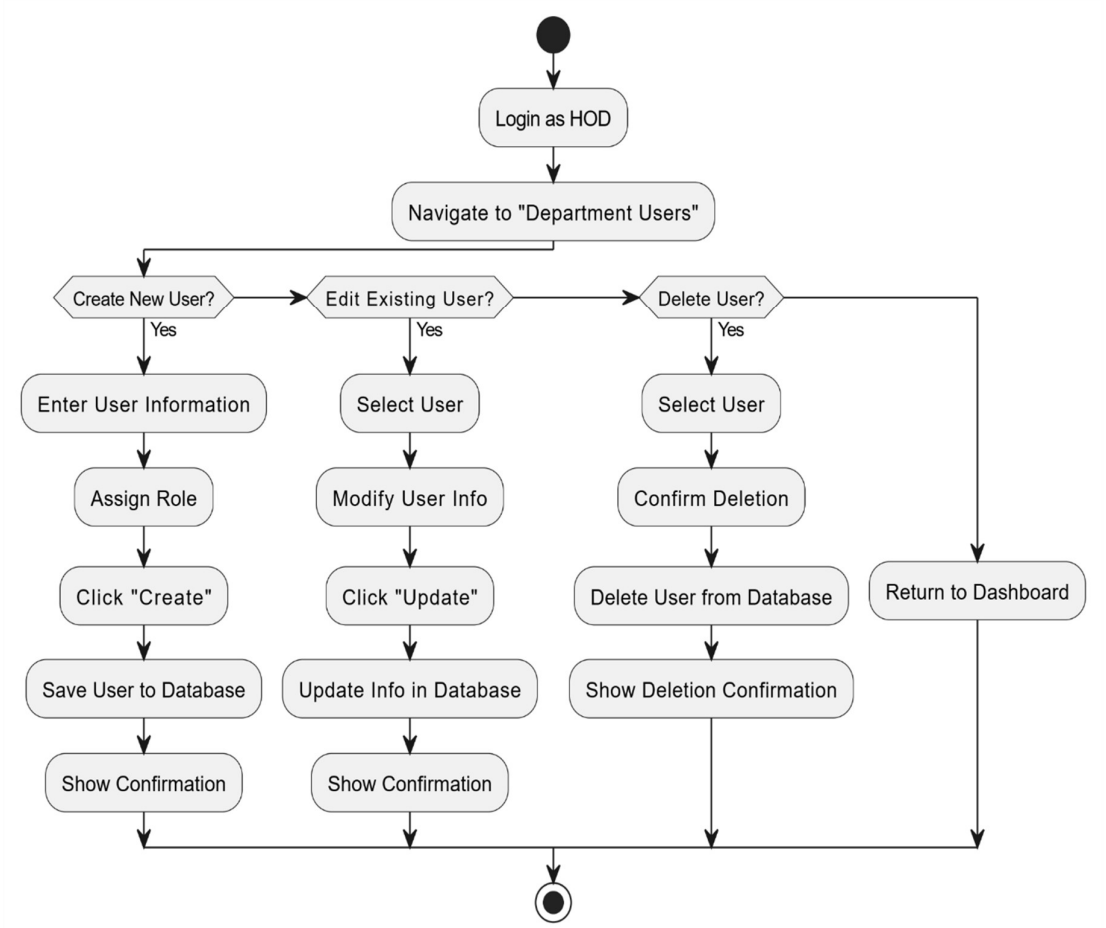

Figure 2.13: HoD Manages Department-Specific User Accounts

Admin Manages All User Accounts : Shows how the admin handles global user account operations

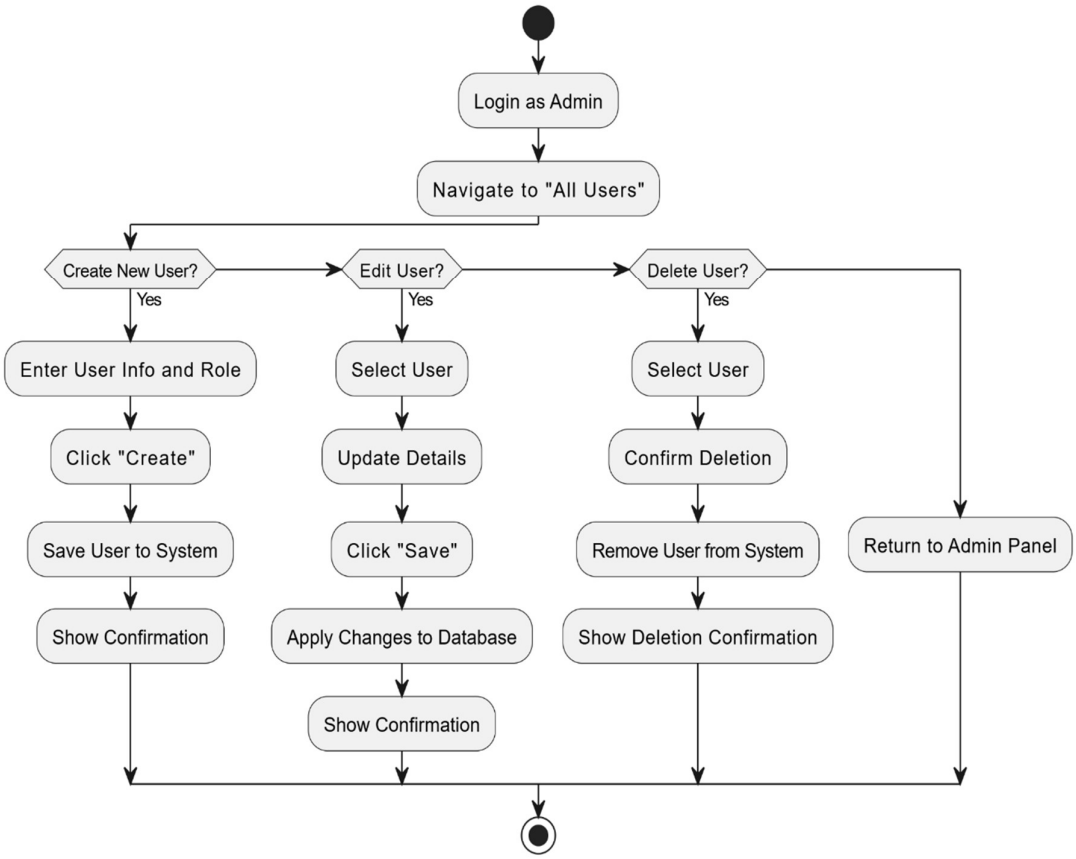

Figure 2.14: Admin Manages All User Accounts

User Edits Profile: Details how users edit their profile information and upload a new photo

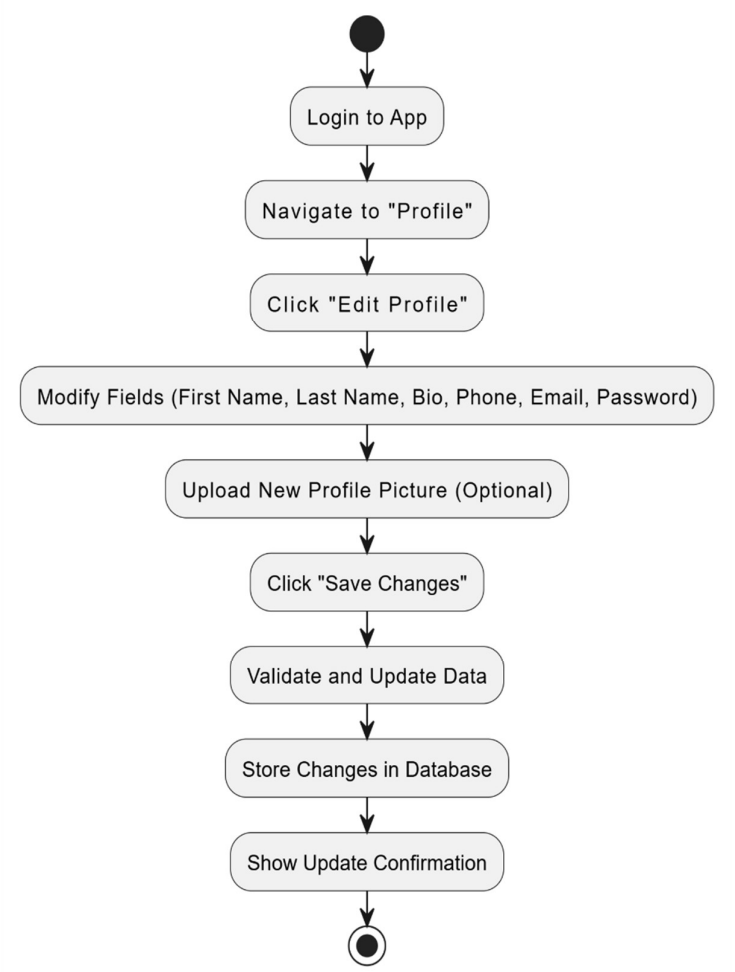

Figure 2.15: User Profile Update

2.3.3 Sequence Diagrams

In this section we use sequence diagrams to represent the chronological interaction between the user and the system, or between different system components during the execution of a specific function, such as login or post creation. These diagrams illustrate the order of messages exchanged between various elements, helping to understand the flow of operations and the coordination between the frontend, backend, and supporting services, to achieve accurate and efficient system design and implementation.

The User Authentication(Login and Logout) : This figure illustrates the login and logout process, where user credentials are verified, access is granted or denied, and the session is terminated upon logout

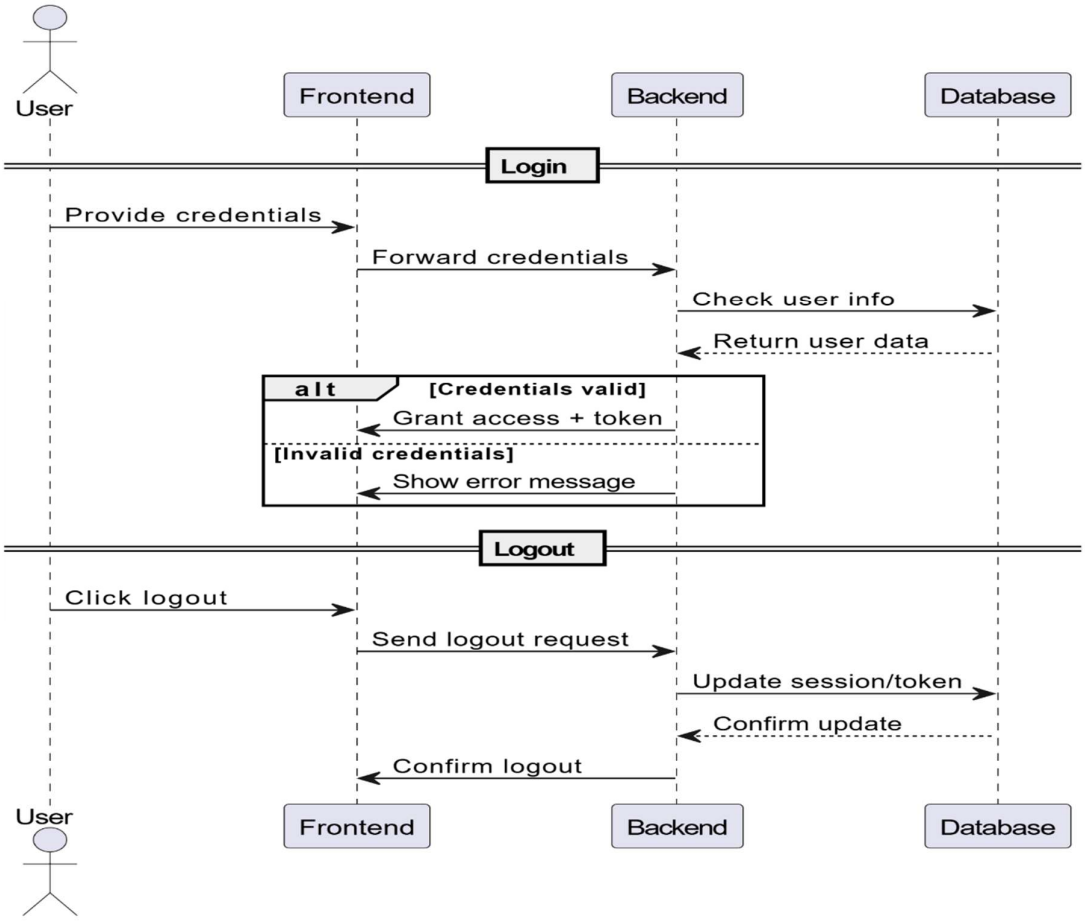

Figure 2.16:User Authentication (Login and Logout)

Teacher Creates a Class and Shares Posts: This figure presents the workflow for a teacher creating a new class and sharing posts. The system stores the data and notifies students accordingly

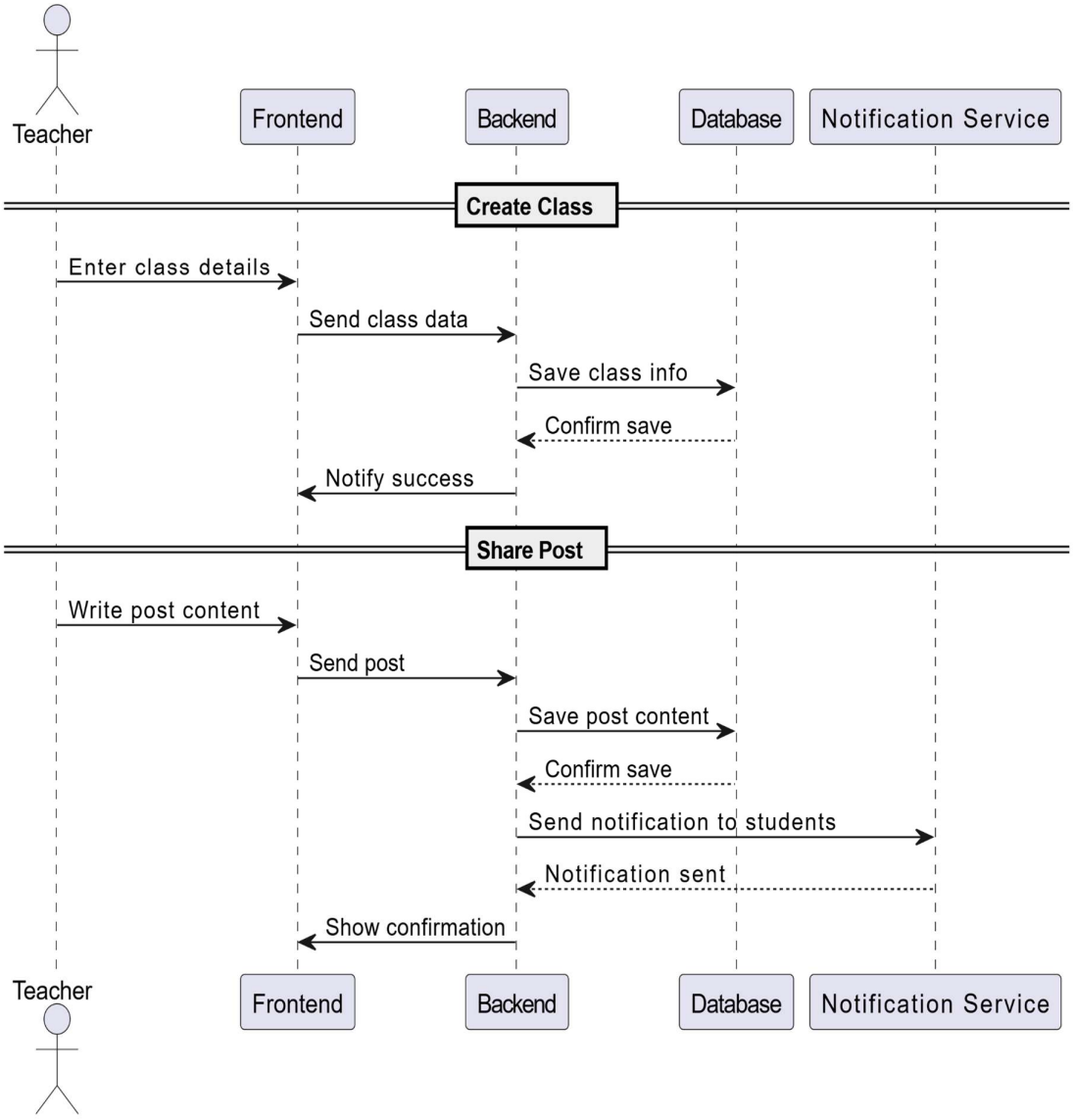

Figure 2.17: Teacher Creates a Class and Shares Posts

Student Enrolls in a Class : This figure shows how a student enrolls in a class. The system processes the request, updates the database, and notifies the teacher

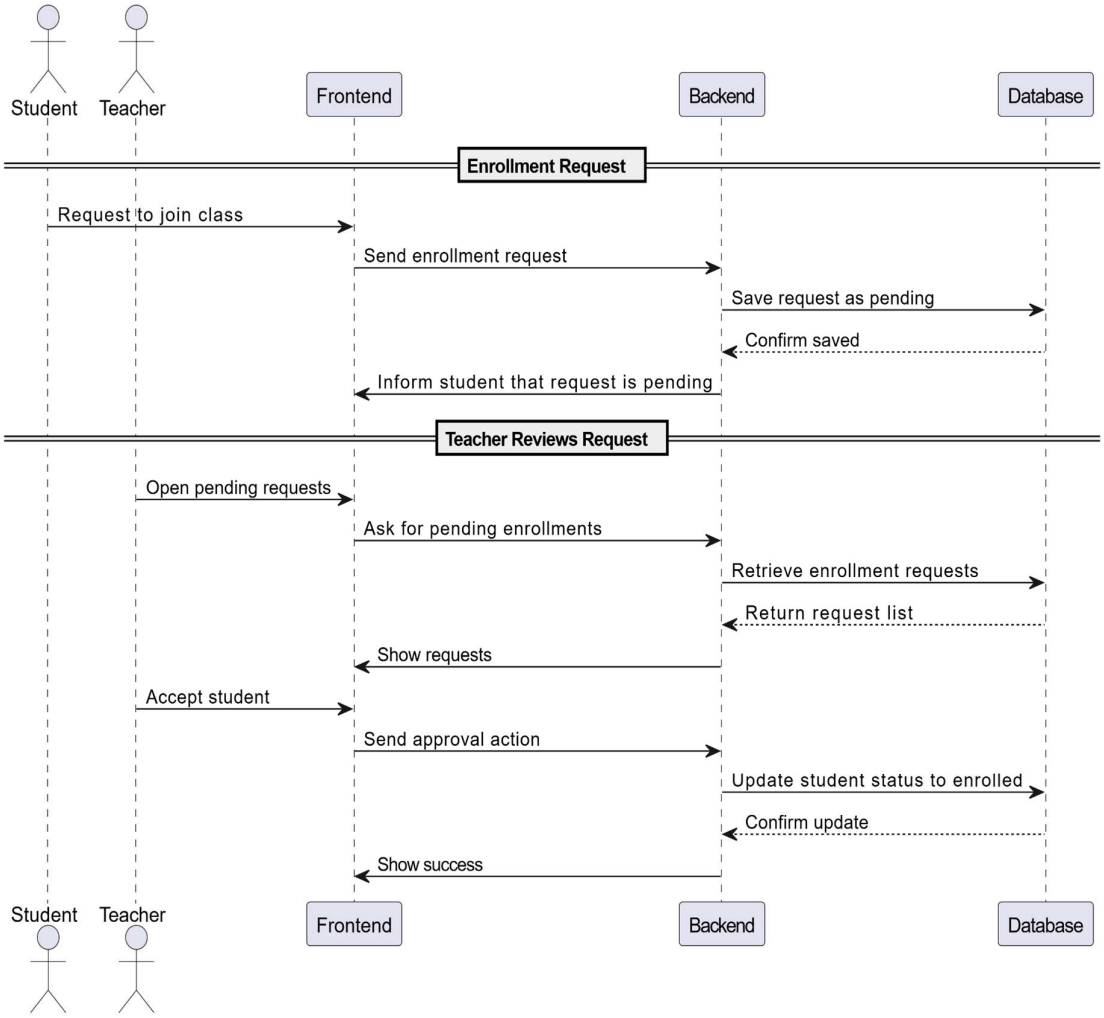

Figure 2.18: Student Enrolls in a Class

Exchange Messages: ,This figure demonstrates the message exchange process between users where messages are stored, notifications are sent, and conversations are retrieved and displayed

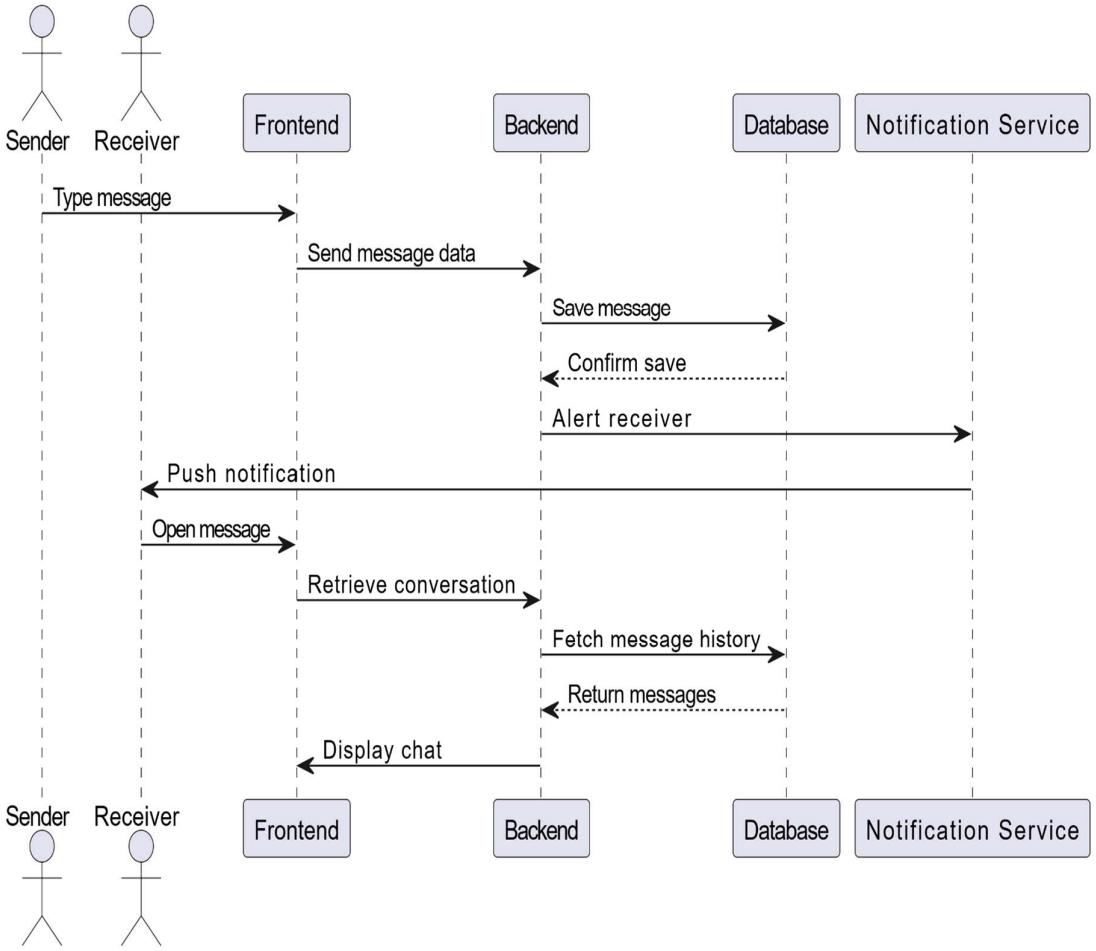

Figure 2.19: Exchange Messages Between Users

2.4 Conclusion

This chapter has provided a detailed exposition of the system design for the ProgNet application, grounded in rigorous architectural and behavioral modeling. The system's internal structure and external interactions have been systematically defined through the application of the C4 model and various UML diagrams. These models ensure alignment with user requirements and facilitate clarity in system logic and component communication. The outcomes of this design phase establish a robust foundation for the subsequent implementation phase, which will be discussed in the next chapter.

Chapter 3

Implementation and Deployment

This chapter presents the implementation and deployment process of the ProgNet system, focusing on the transformation of previously defined design concepts into a fully operational system. We begin by introducing the technologies and tools employed throughout development and deployment, including those related to backend services, authentication mechanisms, and database management. Then, we present the system end-users applications through screenshots, namely the mobile and admin dashboard applications.

3.1 Used Technologies

This section outlines the modern technologies adopted in the development of ProgNet. It covers the development environment, the technology stack employed for both frontend and backend components, the deployment infrastructure, as well as third-party services integrated into the system to enhance its overall functionality, performance, and user experience.

3.1.1 Development Environment

The development environment is composed of the following essential tools:

- Visual Studio Code (VS Code) – is a lightweight, open-source code editor developed by Microsoft. It supports multiple programming languages and offers features like syntax highlighting, debugging, Git integration, and extensions, making it ideal for modern software development [1].
- Postman – is a tool used for testing and validating APIs. It allows developers to send HTTP requests, inspect responses, and debug backend services efficiently, making it an essential part of the API development and testing process [2].
- Git – a distributed version control system, manages source code by saving complete snapshots of the project at each change, enabling efficient tracking, collaboration, and branching throughout the development process [3].

- Github – a cloud-based platform, hosts Git repositories and provides tools for version control, collaboration, issue tracking, and code review, facilitating team-based software development [4].
- Genymotion – a powerful Android emulator for app testing and development, offers fast, reliable performance and supports many virtual devices with different models and Android versions. Provides both cloud and desktop versions and seamless integration with Android Studio and other development tools [5].

3.1.2 Development Stack

Back-end Stack: The backend of our system has been developed using the following technologies:

- Node.js – Open-source, cross-platform runtime environment for running JavaScript outside the browser. Used to build back-end services like APIs for mobile and web apps. Adopted by major companies [6].
- Express.js – Minimal framework built on Node.js to simplify API development. Helps organize application logic using middleware and routing, with utilities for handling HTTP requests and responses [6].
- MySQL – Popular open-source relational database management system developed by Oracle. Manages structured data using SQL (Structured Query Language). Stores data in organized tables with defined relationships. Widely used in mobile and web apps for fast, reliable, and scalable data handling. Supports client/server and embedded systems, and integrates easily with many programming languages [7].
- Sequelize – An ORM (Object-Relational Mapping) tool for Node.js that simplifies interaction with the MySQL database by providing a high-level API for defining models, relationships, and queries [8].

Front-end Stack: The front-end technology stack utilized in this project comprises the following components:

- Flutter – Open-source framework for creating natively compiled, multi-platform mobile applications using a single codebase. Enables fast development with expressive UI and smooth performance across Android and iOS [9].

- Dart – Client-optimized programming language for multi-platform app development, powering Flutter, offering sub-second hot reload, sound typing, and flexible runtimes targeting web, mobile, and desktop [10].
- GetX – popular Flutter library for efficient state management, route management, and dependency injection. Simplifies app logic by reducing boilerplate and enabling reactive programming for smooth mobile app development [11].

3.1.3 Deployment Services

The deployment infrastructure was established on a Virtual Private Server (VPS).

- VPS - is a virtualized server environment hosted on a physical server using virtualization technology. It offers dedicated resources and full administrative access, allowing for customized configurations and better performance compared to shared hosting. ProgNet was deployed on a VPS to ensure stability, security, and scalability during both development and production stages [12].

3.1.4 Third-party Services

To enhance the system's capabilities, ProgNet integrates the following third-party services:

- Firebase Cloud Messaging (FCM) – is a cross-platform cloud messaging service that enables reliable message delivery to client applications. It is commonly used to notify apps of new data availability for synchronization, thereby enhancing user engagement and retention. FCM also supports sending notification messages with a payload of up to 4096 bytes, making it suitable for use cases such as instant messaging [13].
- Supabase – an open-source backend platform used in this system specifically for media storage, such as user profile pictures. Supabase was chosen for its seamless integration with Flutter and efficient handling of image uploads. Meanwhile, MySQL managed the main database, ensuring a flexible and effective separation of data and media services [14].

3.2 Presentation of The Mobile Application and Web Application

In this section we present ProgNet end-users' applications, namely, the mobile application and the admin dashboard web-based application. Both applications are already functional. For the mobile application, it can be downloaded from here [15], whereas the admin dashboard is available through the following link [16].

3.2.1 Mobile Application

Login Page: Provides a secure authentication gateway where users must enter their email and password to access the system. This ensures that only authorized members of the university community can log in. See Figure 3.1

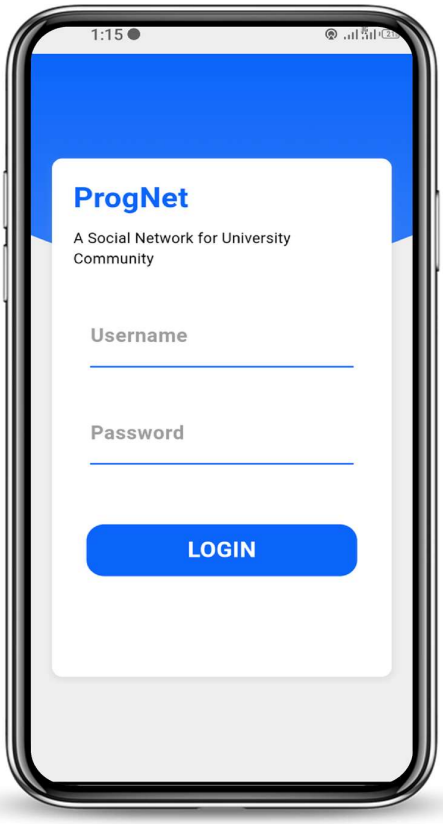

Figure 3.1: Login Page

NewsFeed Page: Provides a centralized space for university, faculty, and department posts, with a private section for departmental announcements managed solely by the Head of Department. See Figure 3.2

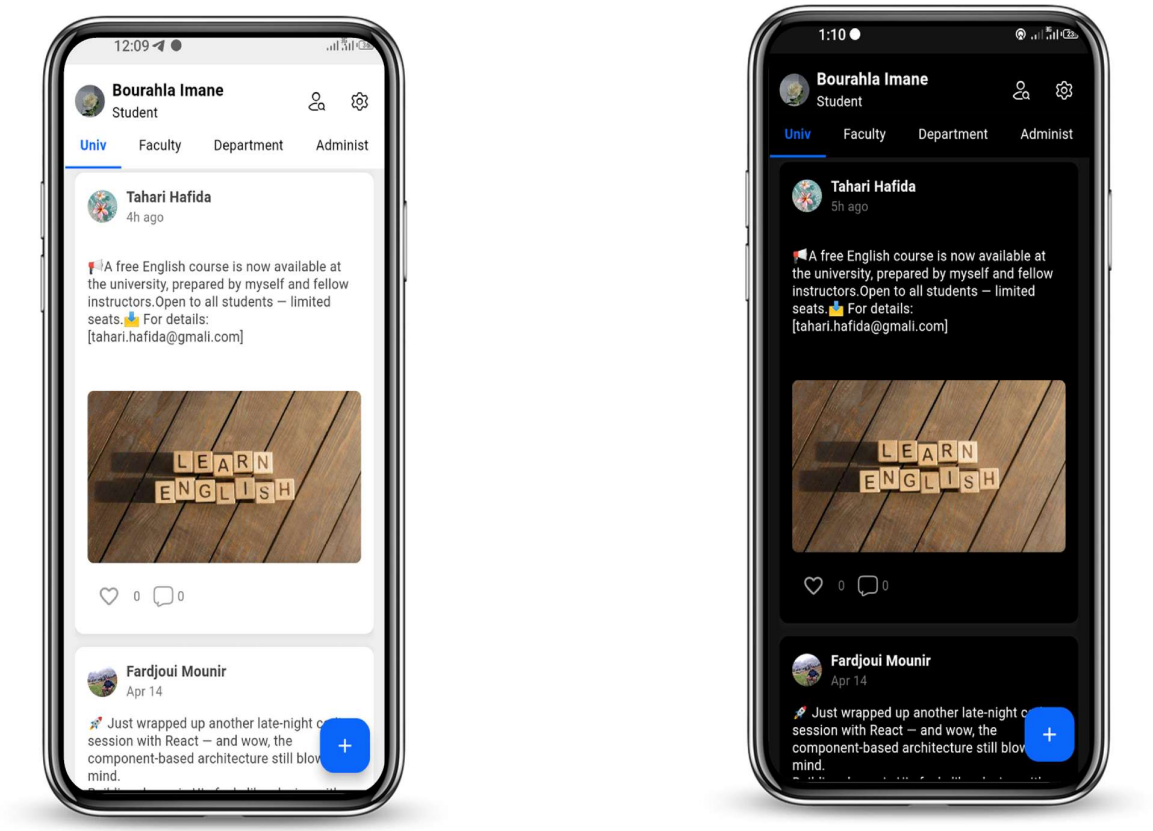

Figure 3.2: NewsFeed Page

Classroom Page: Enables teachers to create and manage classes, while students can request to join and participate once accepted. See Figure 3.3

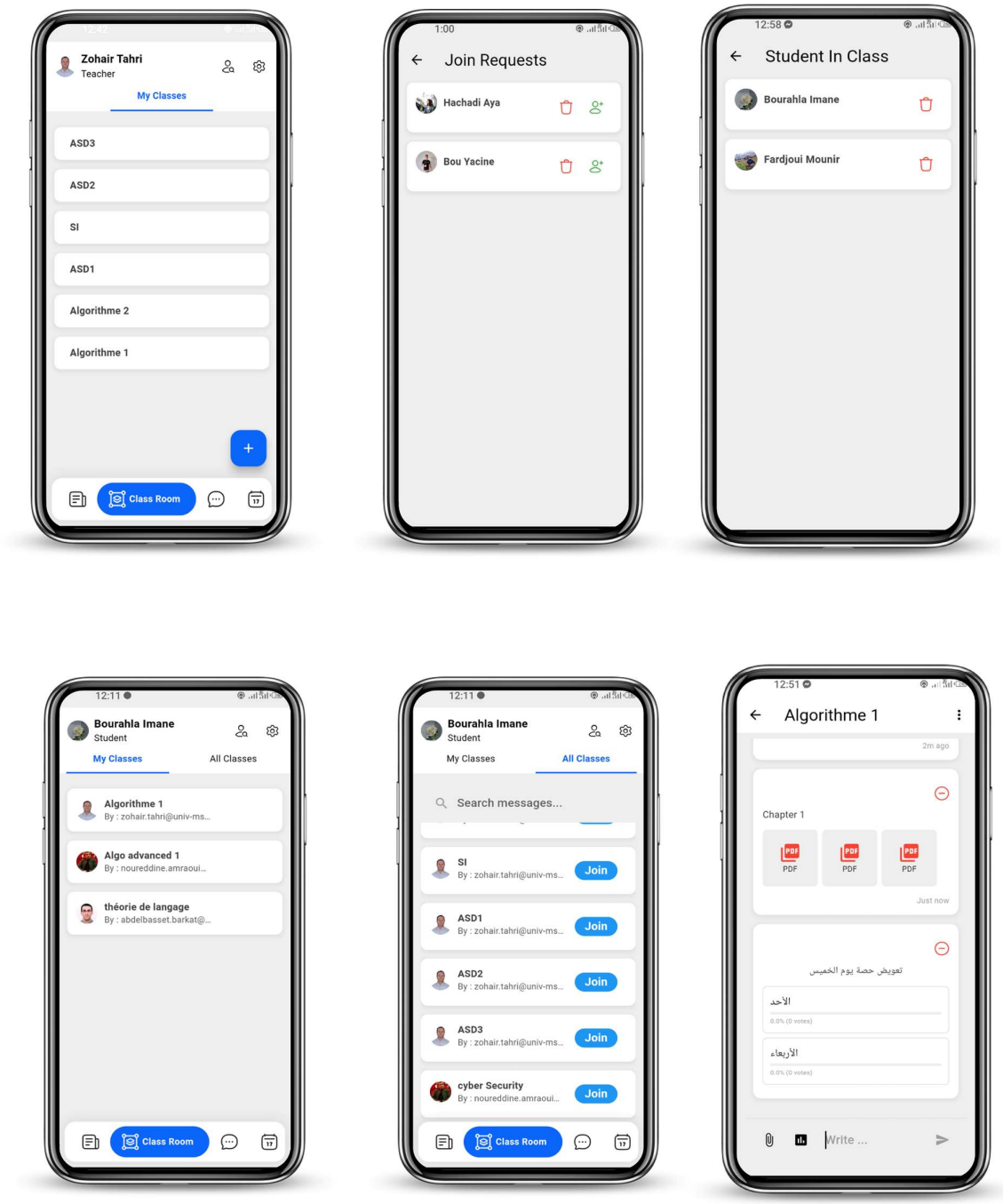

Figure 3.3: Classroom Page

Messages Page: This section provides a secure and private messaging environment that enables users to communicate directly with one another, fostering effective interaction and collaboration within the university community. See Figure 3.4

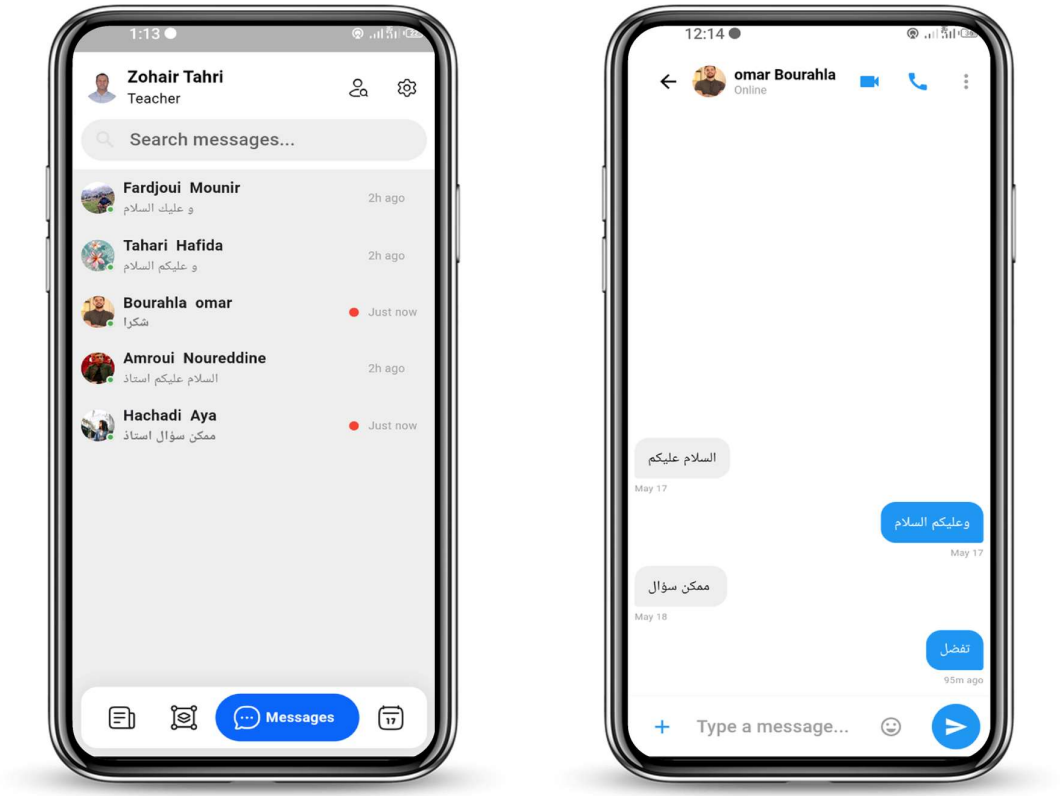

Figure 3.4:Messages Page

Events Page: Allows HoD to post events in the department. Moreover, users in the department can attend the events. See Figure 3.5

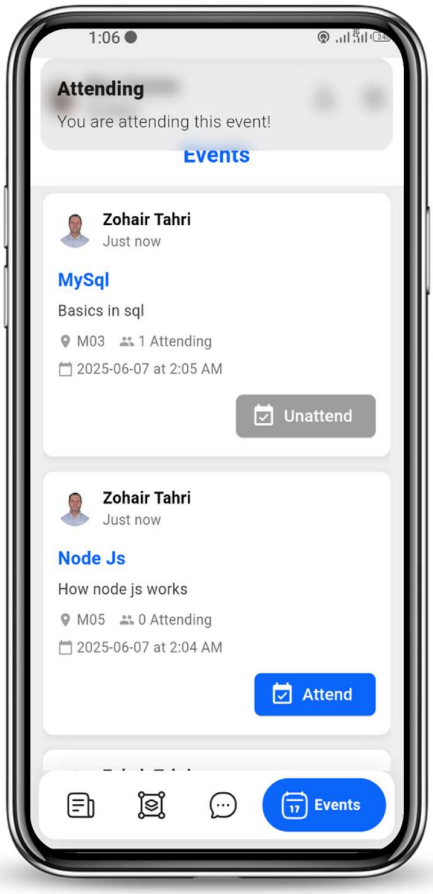

Figure 3.5: Events Page

Settings Page: Users can personalize their experience. They have the option to switch between dark and light mode or the system theme, select their preferred language, and securely log out of the application, and there is a section for HoD dashboard. See Figure 3.6

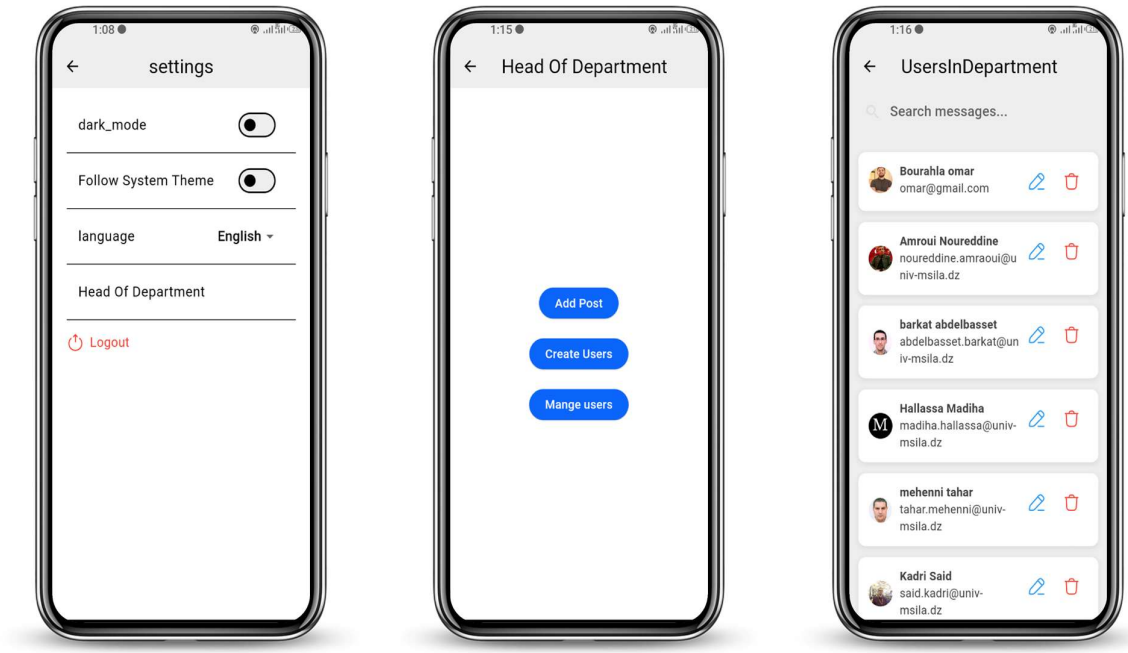

Figure 3.6: Settings Page

Profile Page: This page displays a users personal information and their posts. If the viewer is the profile owner, they can edit their profile otherwise they have the option to send a direct message. See Figure 3.7

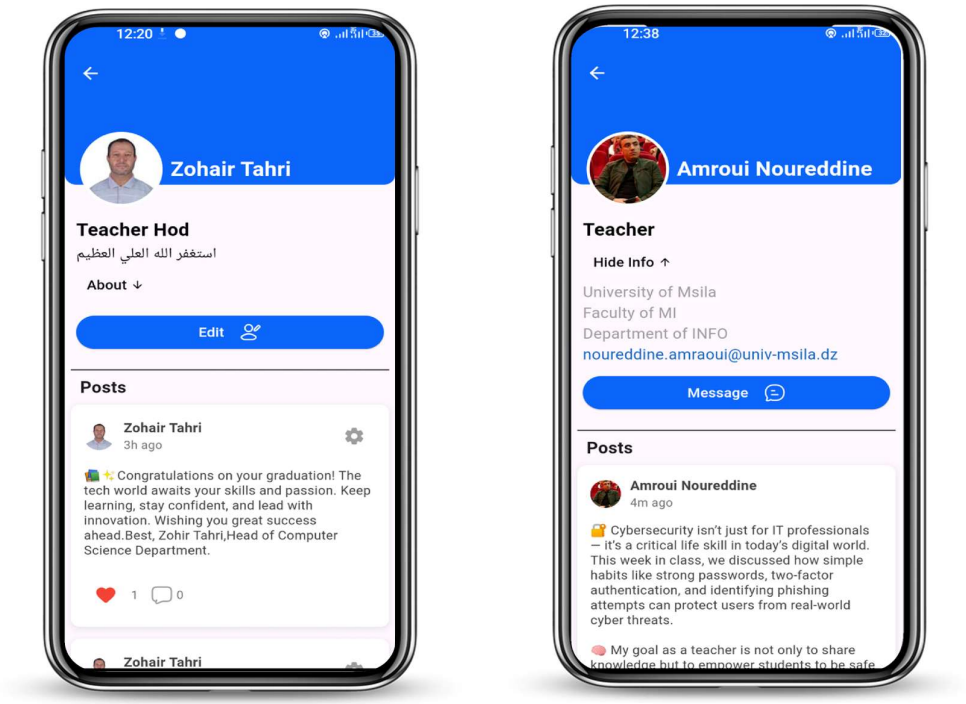

Figure 3.7: Profile Page

User Search Page: Allows users to search for and view profiles of all registered users. This feature facilitates academic and social connections among students, teachers, and staff. See Figure 3.8

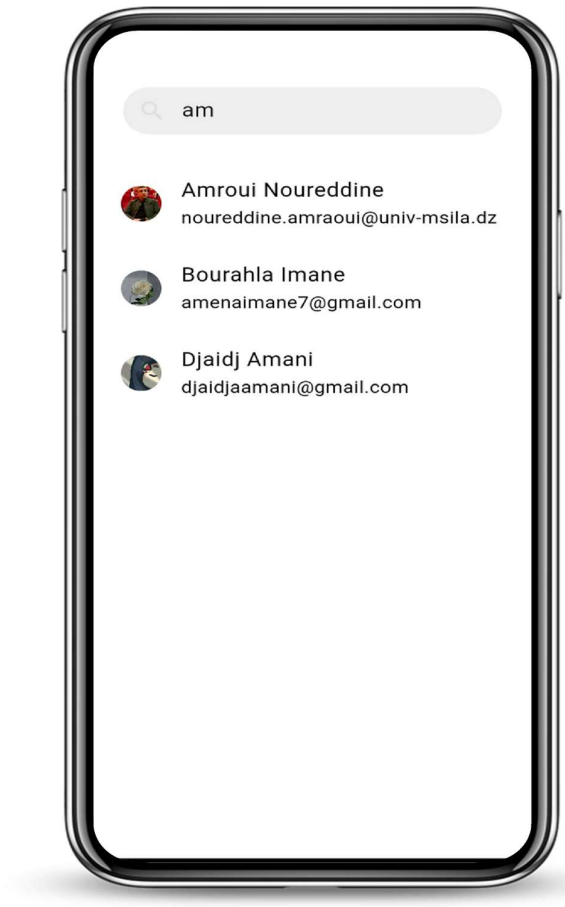

Figure 3.8: User Search Page

3.2.2 Web Application

This application is only dedicated to administrative users, namely the system administrator and the head of department.

Login Page: Secure access requiring email and password for all administrative users. See Figure 3.9

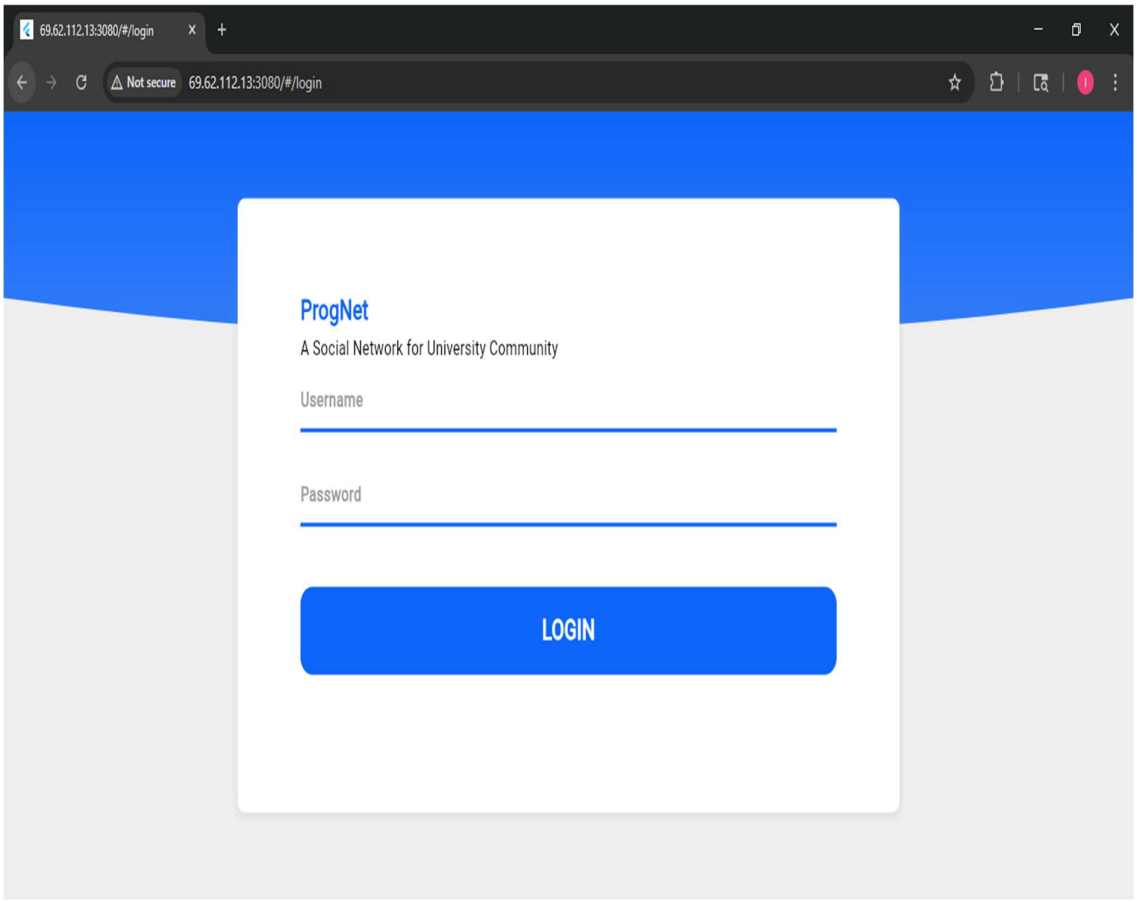

Figure 3.9 :Login Page(in website)

Main Page: After login, users are redirected to a central dashboard tailored to their roles (Admin, Head of Department). This page offers quick navigation to key functions such as user account creation, account management, and role-based administrative tools. See Figure 3.10

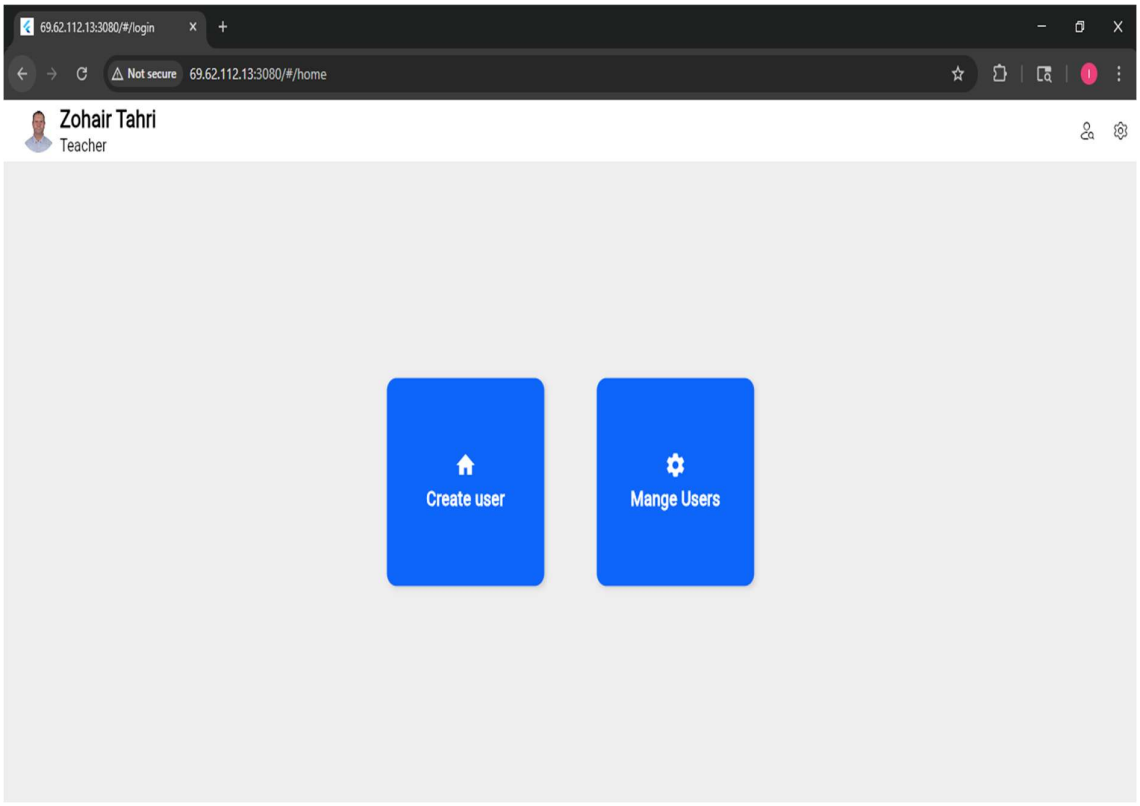

Figure 3.10: Main Page(website)

User Account Creation Page (HoD): Allows authorized administrators to create new user accounts for students, teachers. Each account is assigned to the same department of the HoD, ensuring proper system organization. See Figure 3.11

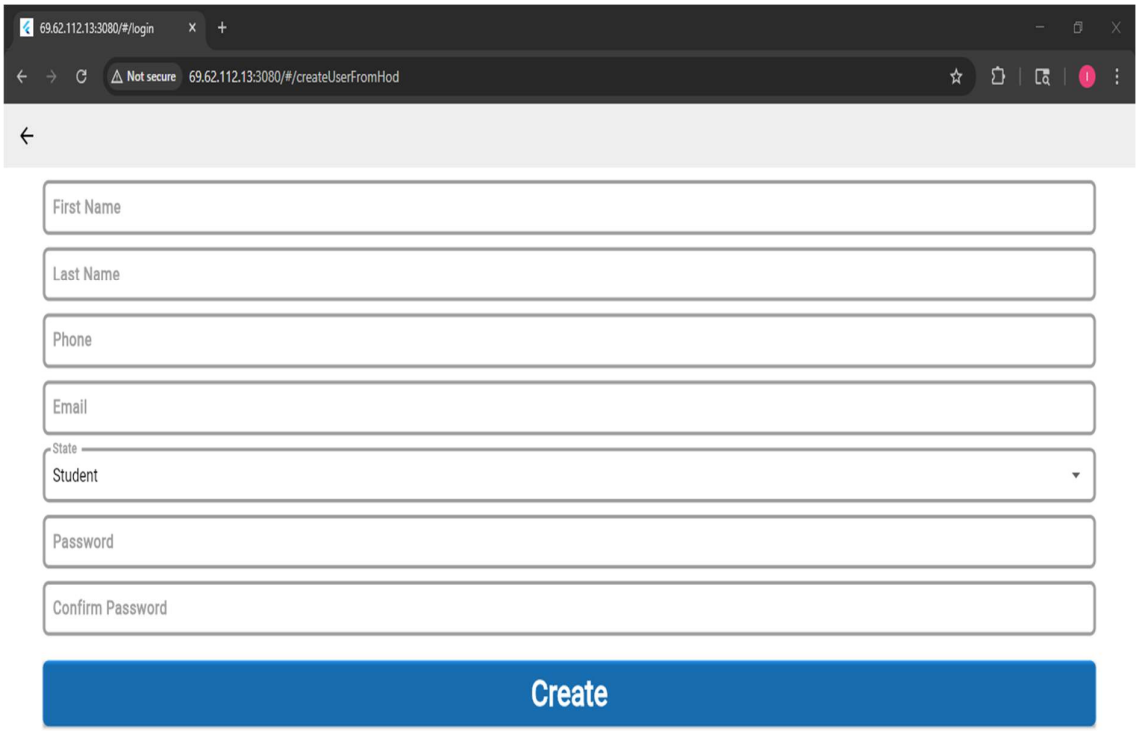

Figure 3.11:User Account Creation Page

User Account Creation Page (Admin): Allows authorized administrators to create new user accounts for students, teachers. Each account is assigned a specific role and department, ensuring proper system organization. See Figure 3.12

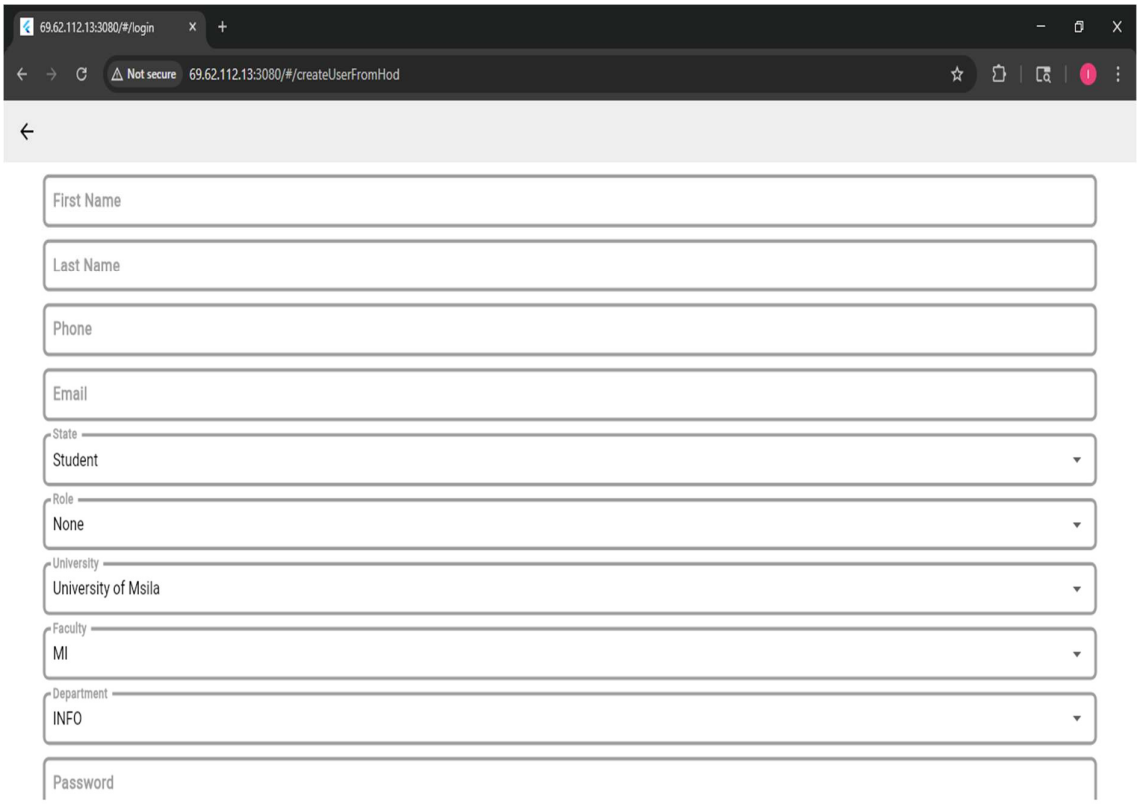

Figure 3.12: User Account Management Page

User Search Page: Provides a robust search function for locating any registered user in the system. Searches can be performed by name, email, facilitating quick administrative access. And HoD can only search for users in his department. See Figure 3.13

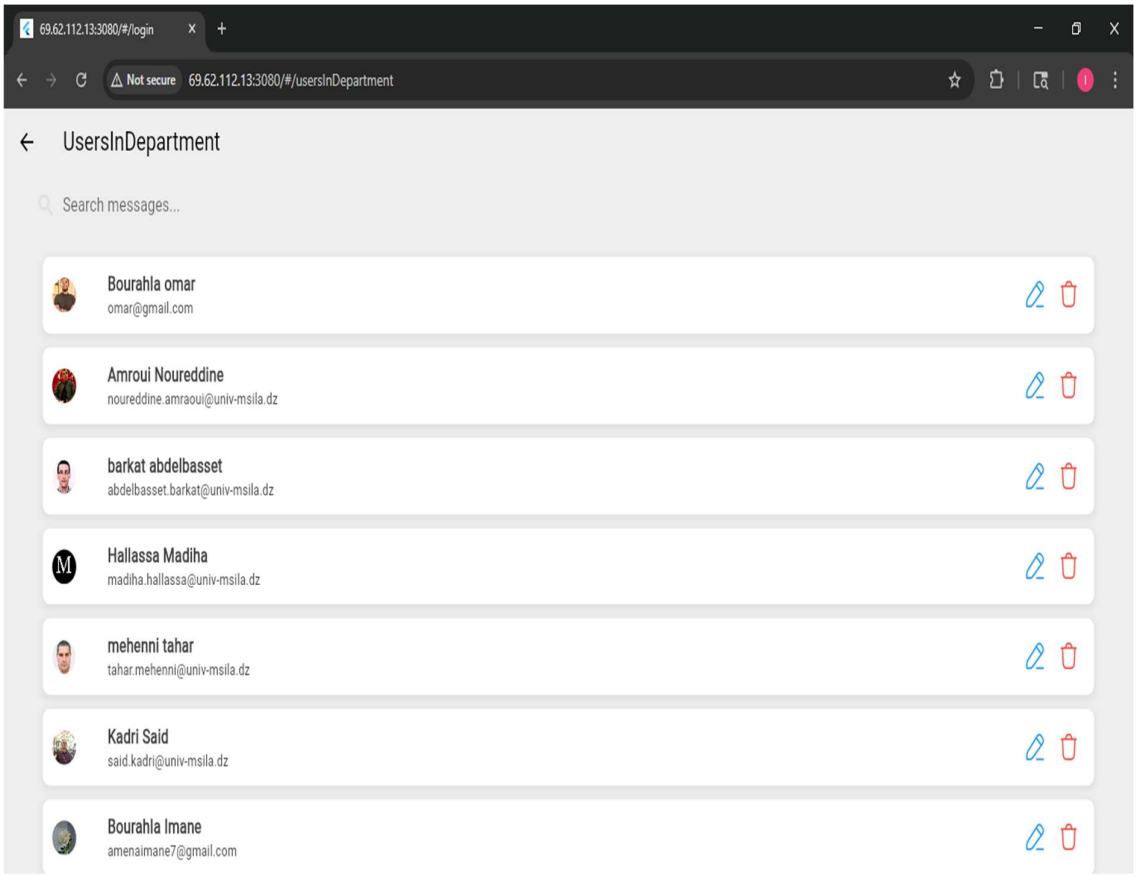

Figure 3.13: User Search Page

User Account Management Page (HoD): Enables HoD to view, edit, or delete existing user accounts enrolled in the same department. They can update user information and reset credentials as needed. See Figure 3.14

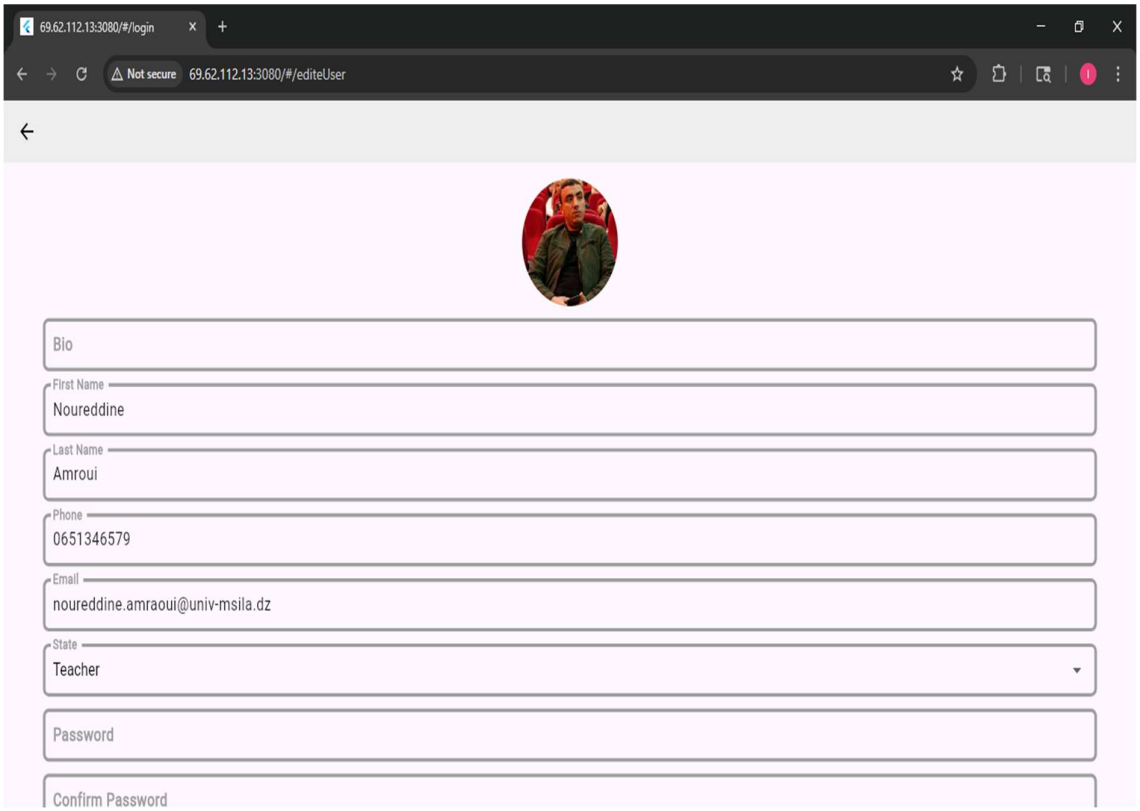

Figure 3.14: Head of Department Page

User Account Management Page (Admin): Enables administrators to view, edit, or delete existing user accounts. They can update user information, change roles and reset credentials as needed. See Figure 3.15

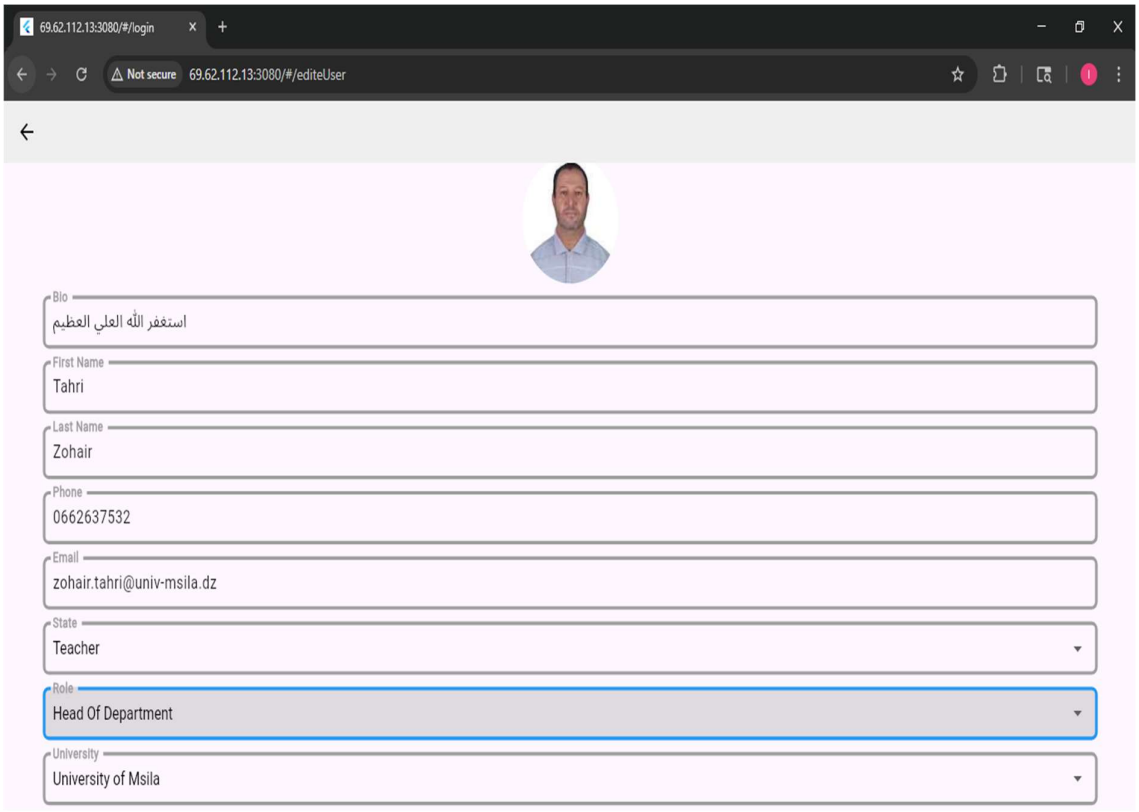

Figure 3.15: Admin Page

3.3 Conclusion

This chapter summarizes the implementation and deployment of the ProgNet system. It presented the technologies and tools used during development, and highlighted the main interface components and their roles. By translating design concepts into a functional system, this phase established the technical foundation for the system, paving the way for performance evaluation and future enhancements.

.

General Conclusion

This report provided a comprehensive documentation of our proposed academic social networking platform, ProgNet, designed specifically for university communities. It covers the entire software engineering lifecycle, from business analysis to system design until implementation and deployment. Leveraging modern technologies such as Flutter, Node.js, Express.js, MySQL, Supabase, and Firebase, we developed a functional system that addresses the shortcomings of existing digital platforms within Algerian universities.

ProgNet facilitates structured interaction among students, faculty, and administrators through features such as user authentication, profile management, class creation, post sharing, event management, private messaging, and administrative oversight. The platform aims to reduce dependency on third-party social media tools while enhancing academic engagement and institutional communication.

Although a wide range of features has been successfully implemented, there are several directions for future enhancement. These include the integration of AI-powered recommendations for content and class enrollment, advanced analytics for user engagement, and interoperability with national systems such as Progres and Moodle.

References

[1] "Visual Studio Code," Visual Studio Code, 21 05 2025. [Online]. Available: https://code.visualstudio.com/.

[2] "Postman," 21 05 2025. [Online]. Available: https://www.postman.com/.

[3] "Git," 21 05 2025. [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F.

[4] "Github," 21 05 2025. [Online]. Available: https://github.com/.

[5] "Genymotion," 21 05 2025. [Online]. Available: https://robotqa.com/blog/exploring-genymotion-and-its-alternatives/.

[6] "Node vs Express | Geeksforgeeks," 21 05 2025. [Online]. Available: https://www.geeksforgeeks.org/node-js-vs-express-js/.

[7] "what-is-mysql," 21 05 2025. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html.

[8] "sequelize," 21 05 2025. [Online]. Available: https://sequelize.org/docs/v6/.

[9] "Flutter," 21 05 2025. [Online]. Available: https://flutter.dev/.

[10] "Dart," 21 05 2025. [Online]. Available: https://dart.dev/overview.

[11] "GetX," 21 05 2025. [Online]. Available: https://prashantv03.medium.com/flutter-getx-state-management-f60d59b5778b.

[12] "Google could," 21 05 2025. [Online]. Available: https://cloud.google.com/learn/what-is-a-virtual-private-server.

[13] "Firebase," 21 05 2025. [Online]. Available: https://firebase.google.com/docs/cloud-messaging.

[14] "Supabase," 21 05 2025. [Online]. Available: https://supabase.com/.

[15] https://tinyurl.com/4kpbhy43

[16] http://69.62.112.13:3080/#/login

Abstract

This report presents ProgNet, a mobile-first academic social networking platform developed to meet the specific communication and collaboration needs of Algerian university communities. ProgNet addresses the limitations of relying on general-purpose platforms such as Facebook for academic interaction by offering a secure, role-based, and university-controlled environment. The system enables students, teachers, to create and manage profiles, share educational content, publish department news, organize events, enroll in classes, and engage in direct communication. Built using modern web and mobile technologies, ProgNet integrates third-party services like Supabase for media management and Firebase for real-time notifications. This technical report documents the full development lifecycle of ProgNet, starting from functional analysis and security requirements definition, through architectural design using C4 and UML models, until system implementation and deployment. The system relies on a flexible backend architecture and an interactive mobile app built with Flutter, ensuring a seamless and efficient user experience.

Keywords: Social Networking, Academic Communication, Digital Platforms, Information Systems, Flutter, Node.js.

ملخص

يقدم هذا التقرير نظام ***ProgNet***، وهو منصة اجتماعية أكاديمية مصممة خصيصًا لتلبية احتياجات التواصل والتعاون داخل الأسرة الجامعية الجزائرية. يهدف النظام إلى معالجة القيود المرتبطة باستخدام منصات التواصل الاجتماعي العامة مثل فيسبوك في السياق الأكاديمي، من خلال توفير بيئة آمنة تعتمد على أدوار المستخدمين وتخضع لإشراف مؤسسات التعليم العالي .يوفر **ProgNet**للطلبة والأساتذة إمكانية إنشاء وإدارة ملفاتهم الشخصية، مشاركة المحتوى التعليمي، نشر الإعلانات والأخبار، تنظيم الفعاليات، الانضمام إلى الأقسام الدراسية، والتواصل المباشر بين المستخدمين. وقد تم تطوير النظام باستخدام تقنيات حديثة في تطوير الويب والتطبيقات المحمولة، مع دمج خدمات خارجية مثل **Supabase**لإدارة الوسائط و **Firebase**لتوفير إشعارات فورية . يوثق هذا التقرير جميع مراحل تطوير النظام بدءًا من التحليل الوظيفي وتحديد المتطلبات الأمنية، مرورًا بتصميم الهيكل المعماري باستخدام نماذج **C4** و**UML**، وصولًا إلى تنفيذ النظام ونشره. يعتمد التطبيق على بنية خلفية مرنة وتطبيق محمول تفاعلي مبني باستخدام **Flutter**، مما يضمن تجربة مستخدم سلسة وفعالة.

الكلمات المفتاحية :الشبكات الاجتماعية، التواصل الأكاديمي، المنصات الرقمية، نظم المعلومات، **Flutter**، **Node.js**