CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: InternMatch

Prepared By: Paul Gilbert Abellana

Date of Submission: February 3. 2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this Software Requirements Specification (SRS) is to provide a clear, detailed, and complete description of the functional and non-functional requirements for the **InternMatch** system.

InternMatch is a web-based internship matching platform designed to connect students seeking internships with organizations offering opportunities.

This document serves as the primary reference for:

- The development team (to guide implementation)
- The project adviser and capstone panel (for evaluation and approval)
- Stakeholders (students, employer representatives, administrators) to understand the system's capabilities and limitations

The intended audience includes project stakeholders, developers, testers, and evaluators.

1.2. Scope

InternMatch provides a centralized web platform where:

Included features (what the system will do):

- Allow users (students, employer representatives, administrators) to register, log in, and manage their accounts with role-based access.
- Enable students to create and manage profiles including academic background, skills, interests, and other relevant information.
- Allow employer representatives to post, edit, and manage internship opportunities (title, description, requirements, field, duration, location, status).
- Permit students to browse, search, filter, and apply to internship postings.
- Support application tracking with statuses (Pending, Accepted, Rejected) visible to both students and employers.
- Provide basic in-app notifications for application status changes (e.g., accepted/rejected).
- Include an admin panel for managing users and monitoring internship postings.
- Use a relational database to store all data securely.

Boundaries / Excluded features:

- No AI-based automated matching or recommendations.
- No resume parsing, automatic upload analysis, or video interview integration.
- No real-time chat, third-party login (Google/Facebook), push/SMS notifications.

- No mobile native application (Android/iOS) — only responsive web design.
- No payroll, stipend management, or advanced analytics/reporting dashboards.
- No external job board integrations.

The system is intended for use by students, employer representatives, and administrators within a university or regional context (e.g., for capstone/OJT purposes).

1.3. Definitions, Acronyms, and Abbreviations

JWT – JSON Web Token (used for secure authentication)

RBAC – Role-Based Access Control (student, employer, admin roles)

MoSCoW – Prioritization method (Must have, Should have, Could have, Won't have)

API – Application Programming Interface (RESTful endpoints in Spring Boot)

PWA – Progressive Web App (optional future enhancement for mobile-like experience)

Bcrypt – Password hashing algorithm used for security

User – Any person interacting with the system (student, employer representative, or admin)

Internship – A temporary work opportunity posted by employers

Application – A student's submission to an internship posting

Dashboard – Personalized page showing relevant data (e.g., applications, postings)

2. Overall Description

2.1. System Perspective

InternMatch addresses the common problem faced by students and organizations in finding and managing internships.

Currently, many students rely on manual processes (email, social media, school postings), while employers struggle with disorganized applicant tracking.

InternMatch fits into the larger context of online recruitment and career development platforms (similar to LinkedIn Jobs or JobStreet but focused specifically on internships).

It provides a centralized, secure, and efficient solution tailored for academic and early-career use cases, especially in the Philippine context (OJT requirements).

The system is a standalone web application with a 3-tier architecture:

3. **Presentation layer**: Responsive React frontend
4. **Application layer**: Spring Boot REST API

5. **Data layer**: Relational database (e.g., MySQL/PostgreSQL)

5.1. User Classes and Characteristics

Students (primary users)

- Age: 18–24 (college students)
- Skills: Basic computer literacy, familiar with web apps
- Needs: Easy profile creation, internship search, application tracking
- Frequency: High during semester starts/ends

Employer Representatives (organization users)

- Age: 25–50 (HR, managers)
- Skills: Moderate to high computer literacy
- Needs: Post internships, review applicants, update statuses
- Frequency: Moderate (seasonal)

System Administrators

- Age: 25+ (school IT staff or project maintainers)
- Skills: High technical knowledge
- Needs: Manage users, monitor postings, enforce rules
- Frequency: Low but critical

Guest Users (unauthenticated visitors)

Can browse public internship listings (read-only)

5.2. Operating Environment

Client side:

- Web browsers: Latest versions of Chrome, Firefox, Edge, Safari
- Devices: Desktop, laptop, tablet, mobile (responsive design)
- Operating systems: Windows 10+, macOS 10.15+, Linux, Android 7+, iOS 13+

Server side:

- Backend: Spring Boot (Java 17+)
- Database: MySQL or PostgreSQL
- Hosting: Local development or cloud (e.g., Render, Railway, Heroku free tier)
- Security: HTTPS (SSL certificate)

Development tools:

6. Frontend: React.js + Vite
7. Backend: IntelliJ IDEA / VS Code + Maven/Gradle

8. Version control: Git (GitHub/GitLab)

8.1. Assumptions and Dependencies

Assumptions:

- Users have access to a stable internet connection.
- Students and employers have basic email accounts for registration.
- All users understand basic web navigation and English (or Filipino) language.
- The system will be used primarily during academic semesters.

Dependencies:

9. External libraries: Spring Boot, Spring Security, Spring Data JPA, JWT library (jjwt), bcrypt (via Spring Security), React Router, Axios.
10. No third-party APIs (e.g., no Google OAuth, no email/SMS services).
11. Database must support relational model and be ACID-compliant.

12. System Features and Functional Requirements

Describe each major feature of the system and its functional requirements.

12.1. Feature 1:

Description: Allows users to register, log in, log out, and access features based on their role (student, employer, admin).

Functional Requirements:

- Users can register with name, email, password, and role selection.
- System validates email uniqueness and password strength.
- Passwords are hashed using bcrypt.
- Upon successful login, a JWT token is generated and returned.
- Protected endpoints require valid JWT in Authorization header.
- Logout clears client-side token storage.
- Role-based access: students see profile & applications, employers see postings & applicants, admins see user management.

12.2. Feature 2:

Description: Students can create, update, and view their profiles with academic and skill information visible to employers.

Functional Requirements:

- Students can add/edit academic program, year level, skills (text/JSON), interests.
- Profile is linked 1:1 to user account.
- Employers can view student profiles when reviewing applications.
- Profile completeness indicator (optional).

- Data stored in relational database.

12.3. Feature 3: Internship Posting and Management

Description: Students can search, filter, and apply to internships.

Functional Requirements:

- Students can browse listings with search/filter by field, skills, location.
- Students can view internship details.
- Students can submit applications to open internships.
- One application per student per internship.
- Application status starts as "Pending".

12.4. Feature 4: Application Tracking and Status Updates

Description: Both students and employers can track application status; employers can update it.

Functional Requirements:

- Students see their applications with statuses (Pending, Accepted, Rejected).
- Employers see applicants for their postings and can update status.
- Status change triggers in-app notification to student.
- Status history logged (optional).

12.5. Feature 5: Admin Panel

Description: Administrators can manage users and monitor system content.

Functional Requirements:

- Admins can view, suspend, or delete user accounts.
- Admins can review/approve/remove internship postings.
- Admins have full access to all data.

13. Non-Functional Requirements

Performance

- API response time $\leq$ 2 seconds (95% of requests)
- Page load time $\leq$ 3 seconds on broadband
- Support at least 100 concurrent users without degradation

Security

- HTTPS for all communications
- JWT authentication for protected endpoints

- Passwords hashed with bcrypt
- Protection against SQL injection (prepared statements/JPA)
- Protection against XSS (input sanitization)
- Role-based access control (RBAC)
- Rate limiting on login/auth endpoints (e.g., 100 req/min per IP)

Usability

- Intuitive interface for first-time users (tasks completable in 5–7 minutes)
- Responsive design for mobile, tablet, desktop
- Clear feedback and error messages
- Touch-friendly elements (≥44x44px targets)
- Basic accessibility (WCAG 2.1 Level A where feasible)

Compatibility

- Browsers: Latest Chrome, Firefox, Edge, Safari
- Screen sizes: Mobile (≥360px), Tablet (≥768px), Desktop
- Operating systems: Windows 10+, macOS 10.15+, common Linux

Reliability

- Database transactions are ACID-compliant
- Basic error handling and logging

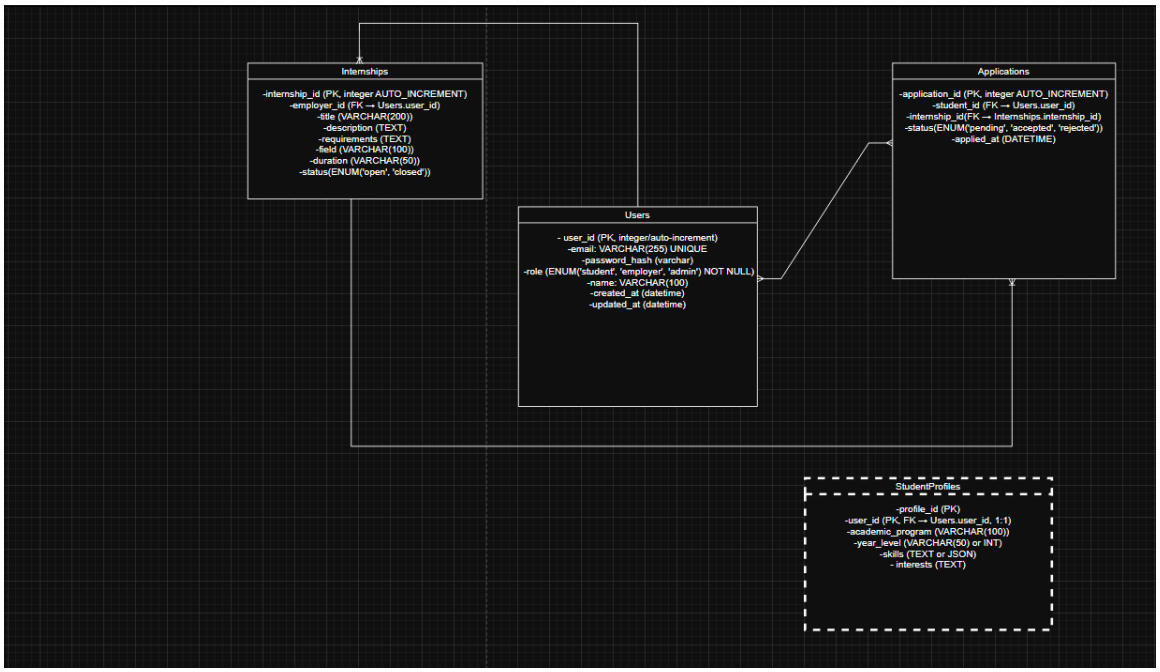
Maintainability

- Clean code structure (MVC pattern)
- Use of Git for version control

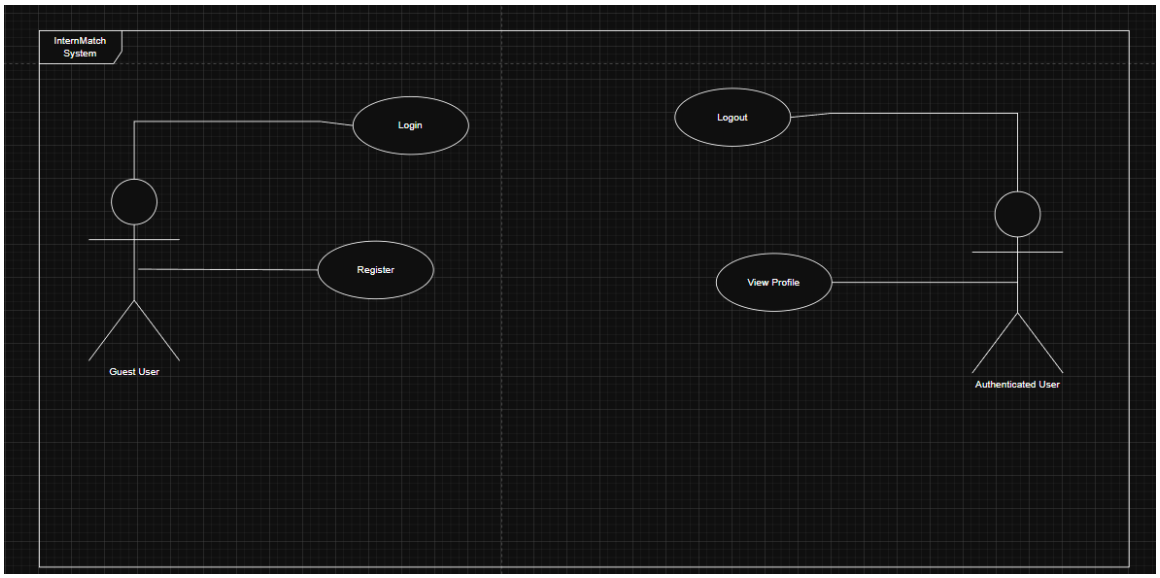
14. System Models (Diagrams)

Insert the necessary diagrams for the system:

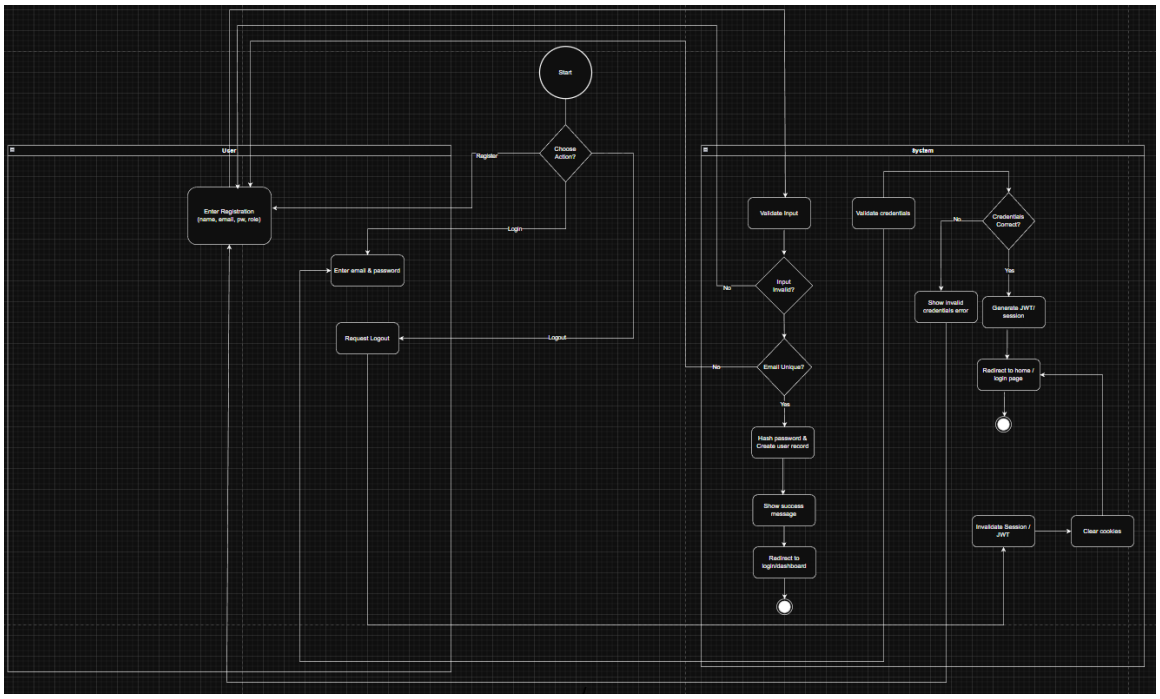
14.1. ERD



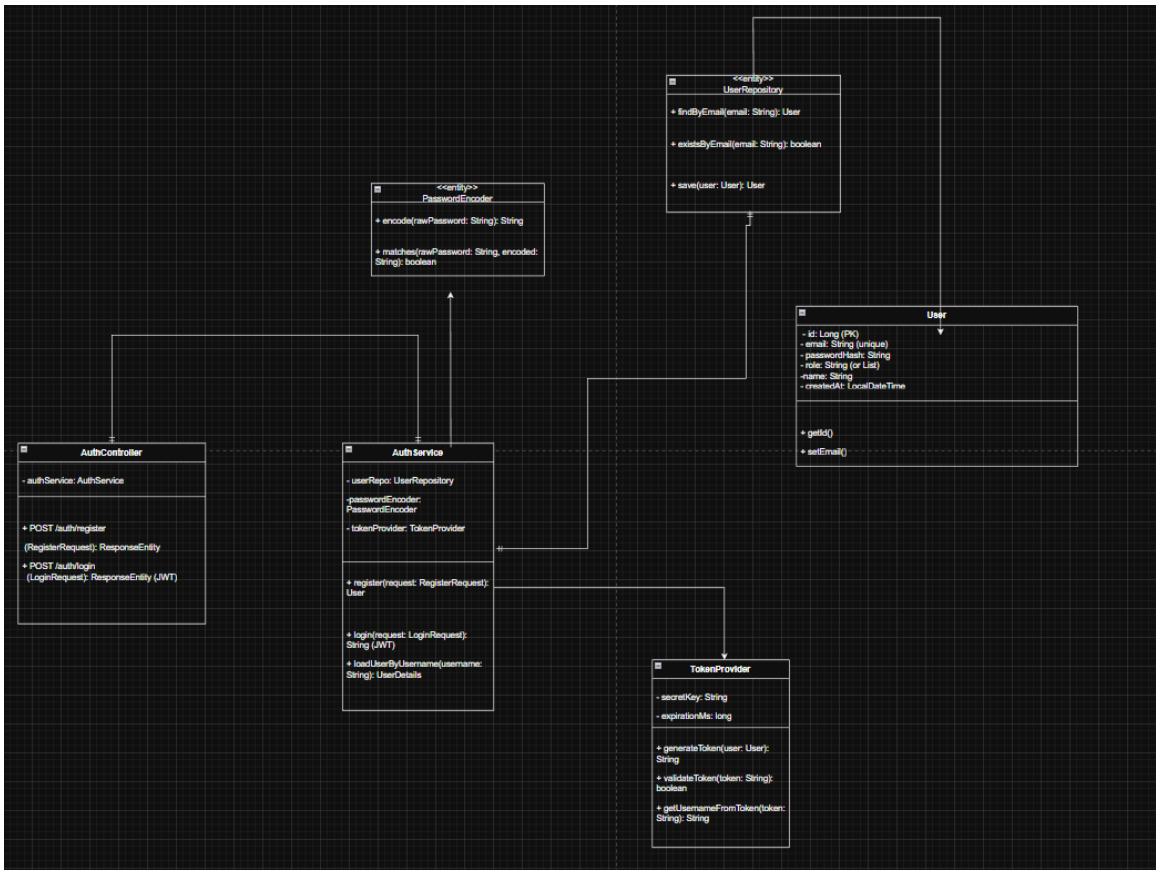
14.2. Use Case Diagram



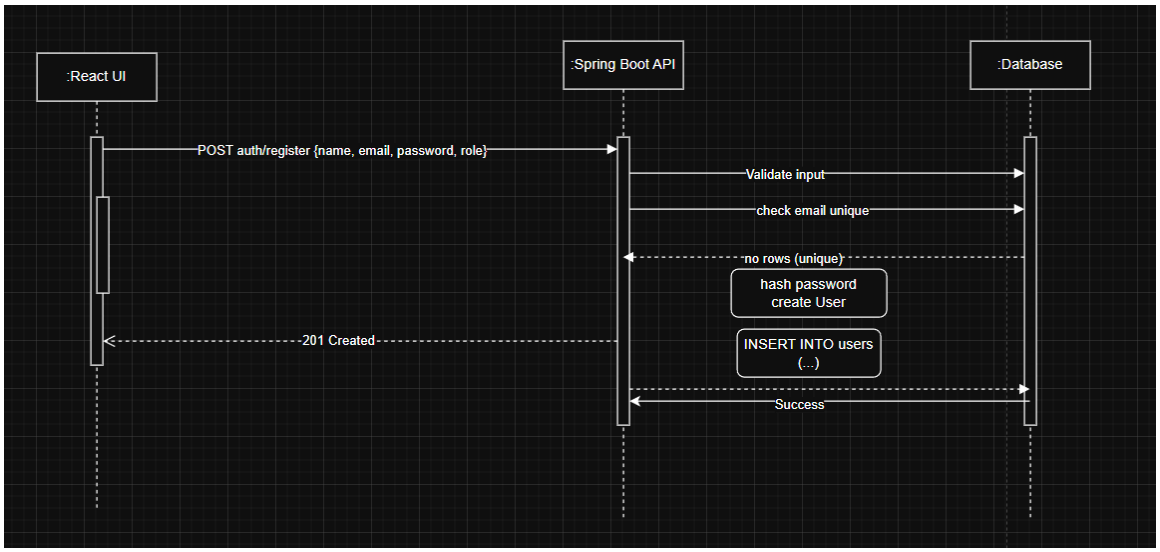
14.3. Activity Diagram



14.4. Class Diagram



14.5. Sequence Diagram



15. Appendices

Appendix A: System Architecture Diagram

Presentation Layer (React UI)

↓ (HTTP/REST + JWT)

Application Layer (Spring Boot API)

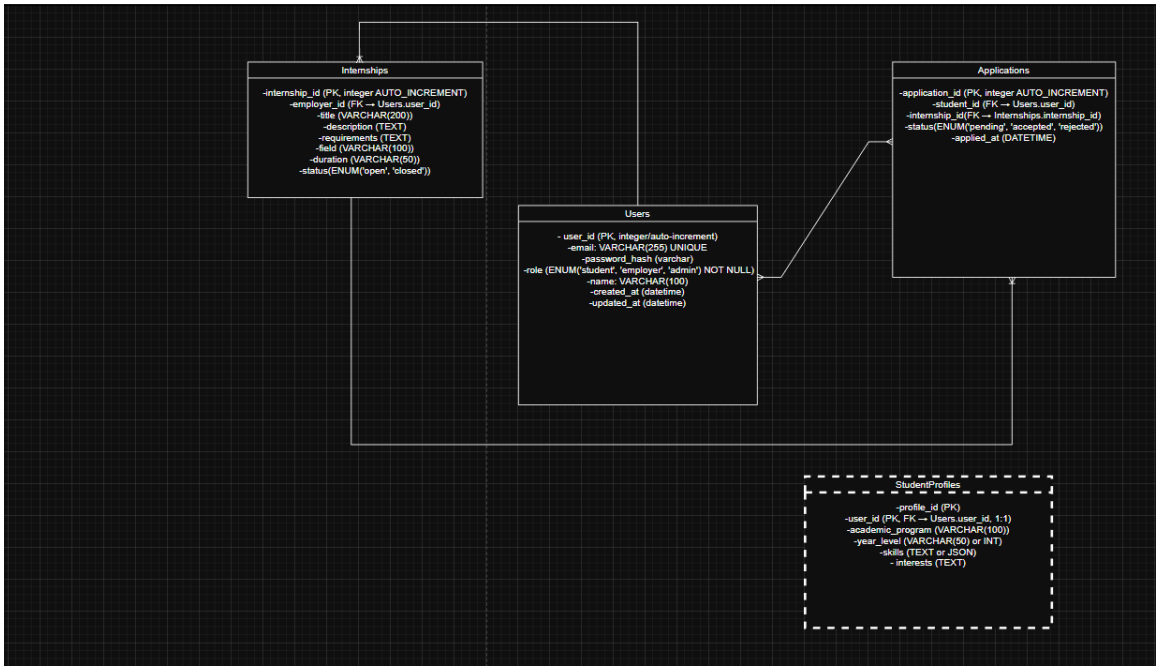
↓ (JPA/Hibernate)

Data Layer (Relational Database)

Description: The system follows a classic 3-tier web architecture:

- **Presentation Tier**: React.js frontend (responsive web UI)
- **Application Tier**: Spring Boot REST API (handles business logic, authentication, JWT)
- **Data Tier**: Relational database (MySQL or PostgreSQL)

Appendix B: Entity-Relationship Diagram (ERD)



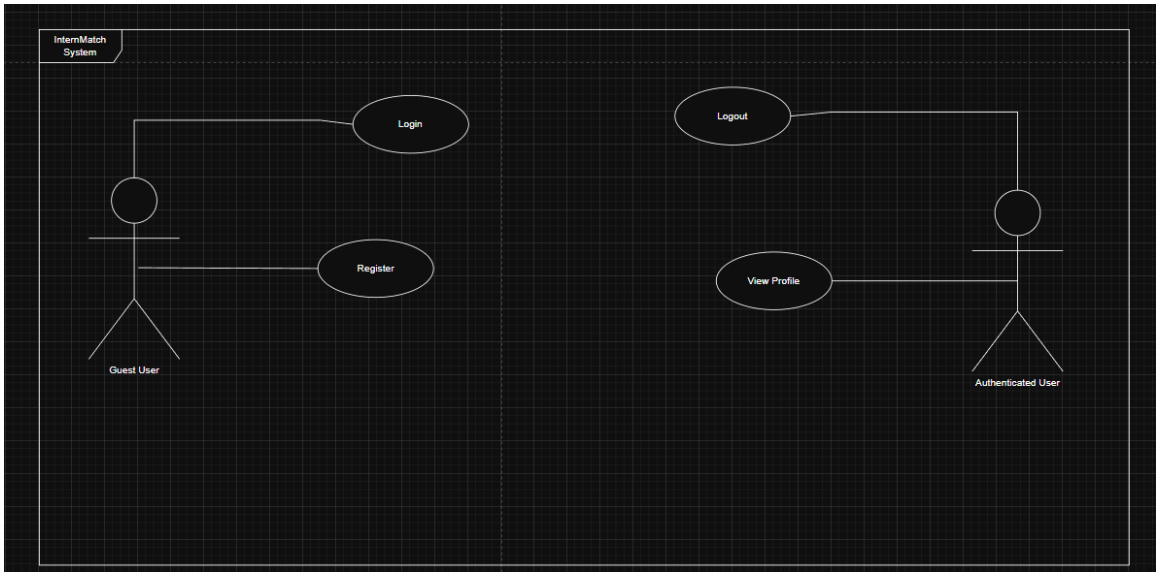
The relational database schema includes the following main entities and relationships:

- Users (strong entity) 1:1 → StudentProfiles (weak entity)
- Users (employer) 1:* → Internships
- Users (student) *:* → Applications (via student_id)
- Internships 1:* → Applications

Key tables:

- Users (user_id PK, email UNIQUE, password_hash, role ENUM, name, timestamps)
- StudentProfiles (user_id PK/FK, academic_program, year_level, skills, interests)
- Internships (internship_id PK, employer_id FK, title, description, status)
- Applications (application_id PK, student_id FK, internship_id FK, status ENUM)

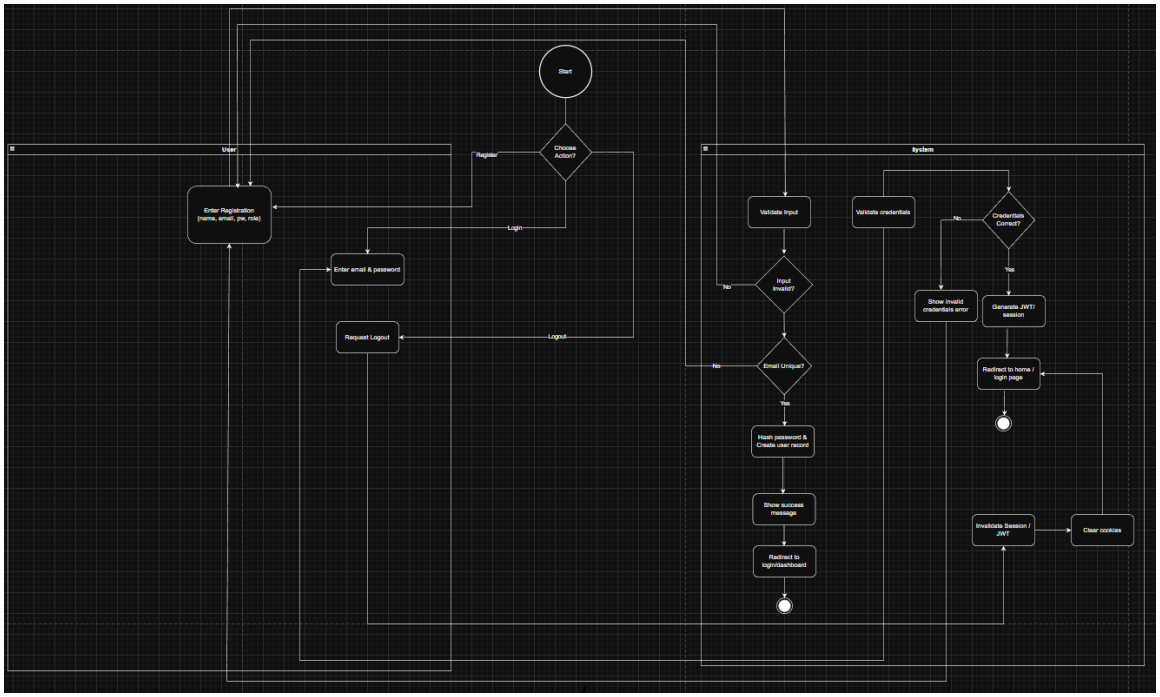
Appendix C: Key Use Case Diagram



Description: Primary actors:

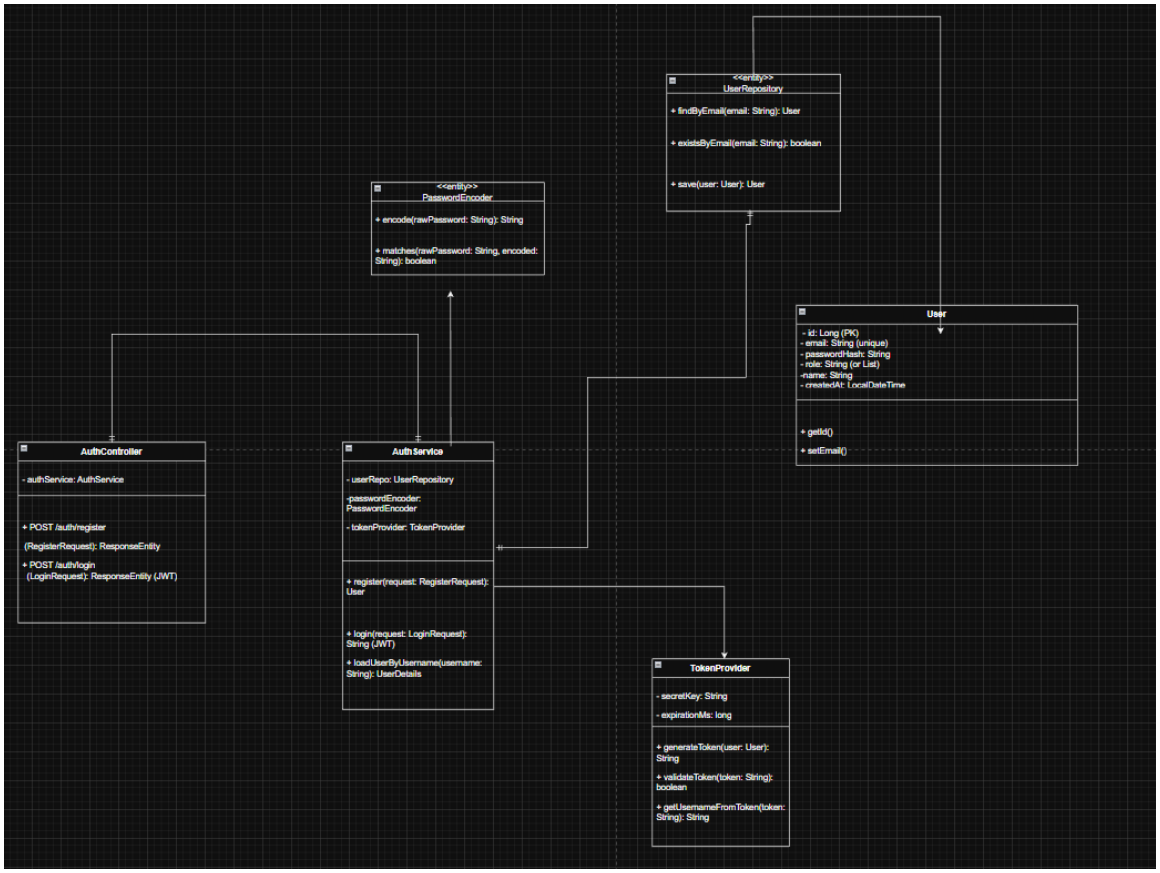
- Guest User → Register, Login
- Authenticated User → Logout, View Profile

Appendix D: Activity Diagram – Authentication Flows



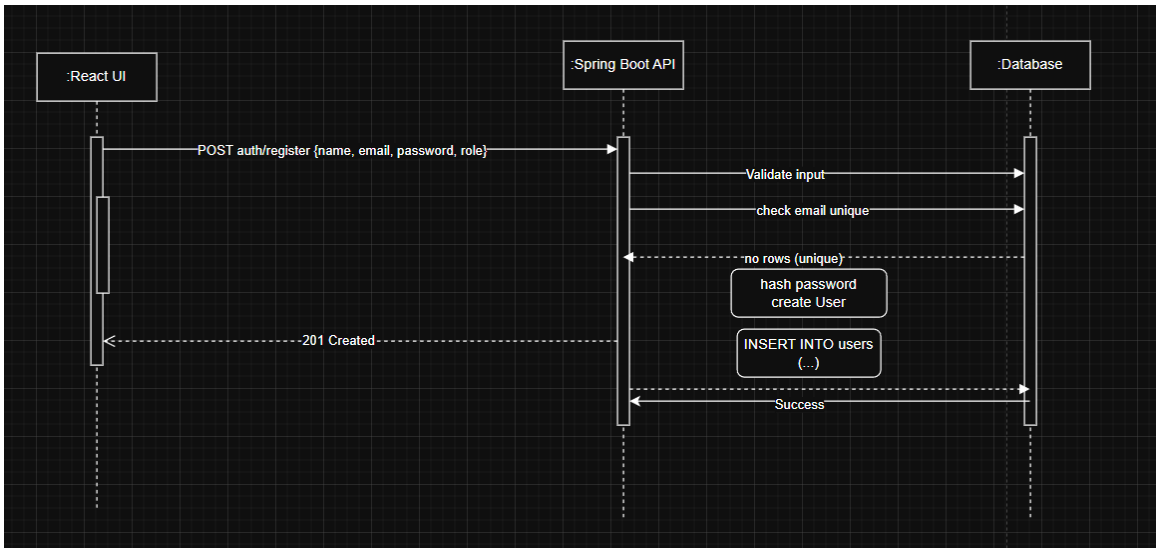
Description: Shows the detailed flow for registration (with validation & uniqueness check), login (credential validation & JWT generation), and logout (session invalidation).

Appendix E: Class Diagram – Authentication Backend



Description: Illustrates the main backend classes involved in authentication and user management using Spring Boot and JWT.

Appendix F: Sequence Diagrams



Register Sequence: React UI → Spring Boot API → Database (validation, uniqueness check, hash & insert)

Login Sequence: React UI → Spring Boot API → Database (find user, validate password, generate JWT)

Logout Sequence: React UI clears token (optional server blacklisting)

Appendix G: References

- Spring Boot Documentation: https://spring.io/projects/spring-boot
- Spring Security JWT Guide: https://spring.io/guides/topicals/spring-security-architecture
- React Documentation: https://react.dev
- IEEE Std 830-1998 – Recommended Practice for Software Requirements Specifications
- GeeksforGeeks – UML Diagrams: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

Appendix H: Glossary (optional, if not covered in 1.3)

- JWT: JSON Web Token – secure way to transmit information as a JSON object
- RBAC: Role-Based Access Control – restricts access based on user roles
- OJT: On-the-Job Training (common Philippine term for internships)