CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G1
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App - User Registration & Authentication

Prepared By: Muriel Pacio

Date of Submission: February 3, 2026

Version: 2

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to provide a complete design specification for the User Registration and Authentication System. This includes detailed system diagrams (ERD, Use Case, Activity, Class, and Sequence diagrams) that will guide the implementation phase.

1.2. Scope

This system will enable users to:

- Create new user accounts with secure credential storage
- Authenticate using username/email and password
- Access protected profile and dashboard pages when authenticated
- Securely logout from the system
- Prevent unauthorized access to protected resources

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
JWT	JSON Web Token - Jason Token-based authentication standard
BCrypt	Password hashing algorithm
SRS	Software Requirements Specification
ERD	Entity Relationship Diagram
REST	Representational State Transfer
API	Application Programming Interface

2. Overall Description

2.1. System Perspective

The User Registration and Authentication System is a full-stack web application consisting of a React-based frontend and a Spring Boot backend with MySQL database. The system implements industry-standard security practices including password hashing with BCrypt and token-based authentication using JWT (JSON Web Tokens).

2.2. User Classes and Characteristics

User class	Characteristics
Unauthenticated user	• Not logged into the system • No personal data stored • Limited access to public pages • Cannot access protected resources
Authenticated user	• Successfully logged in • Valid JWT token present • Personal profile exists • Active session maintained

2.3. Operating Environment

2.3.1. Hardware Requirements

Component	Minimum Specification	Recommended Specification
Development Machine	• Processor: Intel Core i3 or equivalent • RAM: 4GB • Storage: 20GB free space	• Processor: Intel Core i5 or higher • RAM: 8GB or more • Storage: 50GB SSD
Server (Production)	• Processor: 2 cores • RAM: 2GB • Storage: 20GB • Network: 100 Mbps	• Processor: 4+ cores • RAM: 4GB or more • Storage: 50GB SSD • Network: 1 Gbps
Client Device	• Any modern device with web browser • Screen resolution: 1024×768 • Internet connection	• Desktop / Laptop / Tablet / Mobile • Screen resolution: 1920×1080 • Stable internet connection

2.3.2. Software Requirements

Software Type	Component	Details
Operating System	Development	• Windows 10/11 • macOS 10.15+ • Linux (Ubuntu

		20.04+)
	Server	• Linux (Ubuntu 20.04+ recommended) • Windows Server • Docker container
Runtime Environment	Java Development Kit (JDK)	• JDK 17 or higher
	Node.js	• Version 18.x or higher • npm 9.x or higher
Database	MySQL Server	• Version 8.0 or higher
Web Browser	Client Access	• Chrome 90+ • Firefox 88+ • Safari 14+ • Edge 90+
Application Server	Spring Boot	• Embedded Tomcat (included) • Port 8080 (default)
Web Server	React Development Server	• Port 3000 (default) • Production: Nginx or Apache

2.3.3. Development Tools

Tool Category	Tool Name	• Purpose
IDE	IntelliJ IDEA / Eclipse / VS Code	Java / Spring Boot development
	VS Code / WebStorm	React development
Build Tool	Maven / Gradle	Java project build and dependency management
	npm / yarn	React package management
Version Control	Git	Source code version control

API Testing	Postman / Insomnia	REST API testing and debugging
Database Client	MySQL Workbench / DBeaver	Database management and queries
Browser DevTools	Chrome DevTools	Frontend debugging and network inspection

2.4. Assumptions and Dependencies

2.4.1. Assumptions

Assumption	Impact if Invalid
A-1 Users have access to a modern web browser with JavaScript enabled	System will not function; UI components require JavaScript
A-2 Users have stable internet connection during registration and login	API calls may fail; user experience degraded
A-3 MySQL database server is available and accessible	Application cannot store or retrieve user data
A-4 Users provide unique usernames and email addresses during registration	Validation will reject duplicate entries; user must choose different credentials
A-5 Server has sufficient resources to handle expected user load	Performance degradation or service interruption
A-6 Users understand basic password security practices	Weak passwords may be created; additional validation may be needed
A-7 System operates in a trusted network environment during development	Security vulnerabilities may be exploited in production without HTTPS
A-8 Users do not share their authentication credentials	Unauthorized access to accounts; security breach

2.4.2. Dependencies

Dependency Type	Component	Version / Details
Backend Framework	Spring Boot	• Version 3.x • Spring Security • Spring Data JPA

Frontend Library	React	• Version 18.x • React Router • Axios for HTTP requests
Database	MySQL	• Version 8.0+ • JDBC Driver
Security	JWT Library	• jjwt (Java JWT) • Token generation and validation
	Password Hashing	• BCrypt • Provided by Spring Security
Build Tools	Maven / Gradle	• Dependency management • Project building
	npm	• React package management
Network	HTTP / HTTPS Protocol	• Communication between client and server
Runtime	Java Virtual Machine (JVM)	• JDK 17+ • Executes Spring Boot application
	Node.js Runtime	• For React development server

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description: The system shall allow new users to create accounts by providing username, email, password, first name, and last name. All fields are mandatory.

3.2. Feature 2: Input Validation

Description: The system shall validate all input fields for proper format, length constraints, and uniqueness requirements (username and email must be unique).

3.3. Feature 3: Password Security

Description: The system shall hash passwords using BCrypt algorithm with appropriate salt rounds before storing them in the database. Plain text passwords shall never be stored.

3.4. Feature 4: User Login

Description: The system shall authenticate users using their username or email combined with their password. Invalid credentials shall result in an error message.

3.5. Feature 5: Token Generation

Description: The system shall generate a JWT token upon successful authentication containing user identification and expiration information.

3.6. Feature 6: Profile Access

Description: The system shall allow authenticated users to view their profile information and access the dashboard using a valid JWT token.

3.7. Feature 7: Access Control

Description: The system shall prevent unauthenticated users from accessing protected pages and API endpoints. Requests without valid tokens shall be rejected with appropriate HTTP status codes.

3.8. Feature 8: User Logout

Description: The system shall allow users to logout, which clears the authentication token from the client side and invalidates the session.

3.9. Feature 9: Session Management

Description: The system shall maintain user session state using JWT tokens stored in the browser's local storage with appropriate expiration times.

3.10. Feature 10: Error Handling

Description: The system shall provide clear, user-friendly error messages for invalid operations such as duplicate registration, invalid credentials, or expired tokens.

4. Non-Functional Requirements

NFR-1: Security All passwords must be hashed using BCrypt with a minimum work factor of 10. JWT tokens must expire after 24 hours. HTTPS must be used in production.

NFR-2: Performance The system shall respond to user requests within 2 seconds under normal load conditions. Database queries must be optimized with appropriate indexes.

NFR-3: Usability The user interface shall be intuitive and responsive. Error messages shall be clear and actionable. Forms shall provide real-time validation feedback.

NFR-4: Reliability The system shall have 99% uptime. All critical operations must include proper error handling and logging.

NFR-5: Maintainability Code shall follow industry best practices and design patterns. Documentation shall be kept up-to-date. The architecture shall support easy feature additions.

NFR-6: Scalability The system architecture shall support horizontal scaling to handle increased user load without significant code changes.

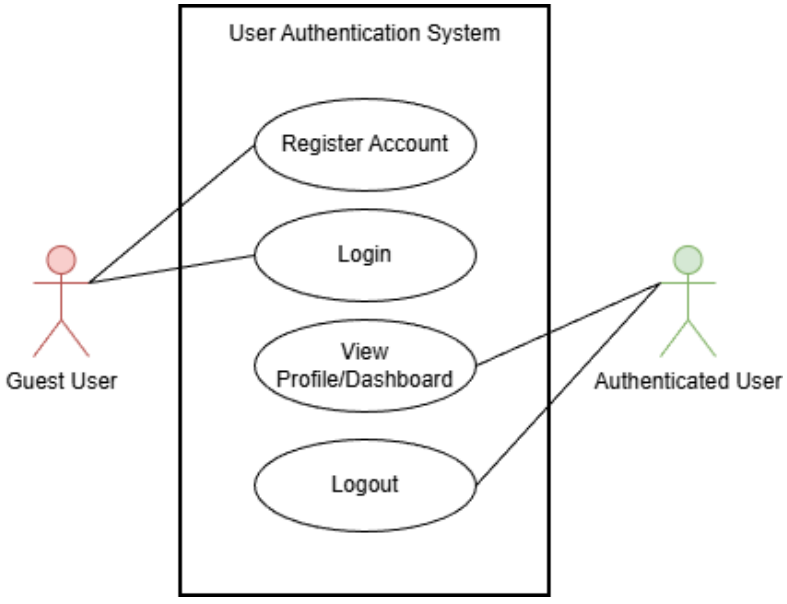
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

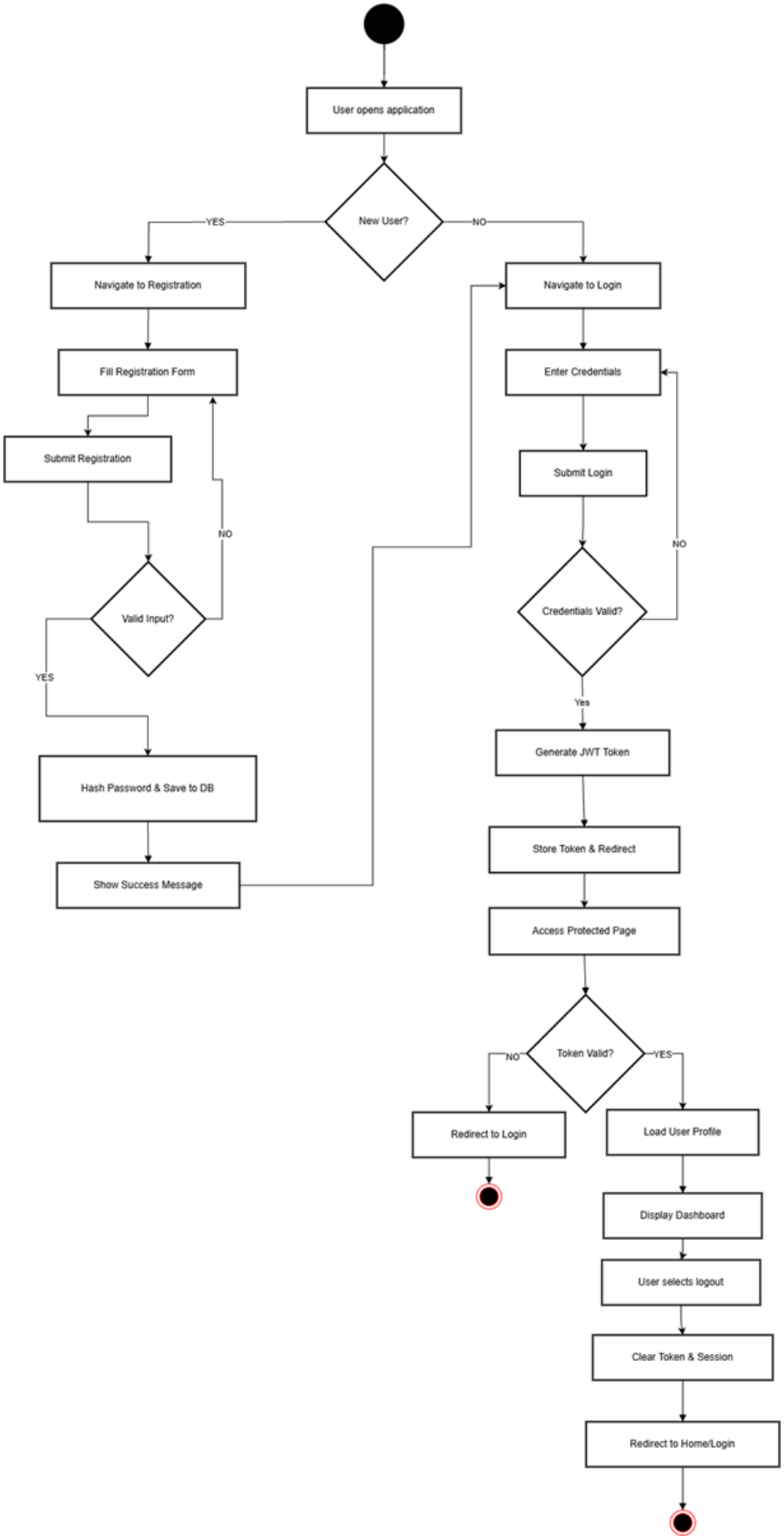
5.1. ERD

USERS		
PK	user_id	BIGINT(20)
	username	VARCHAR(255)
	email	VARCHAR(255)
	password_hash	VARCHAR(255)
	first_name	VARCHAR(255)
	last_name	VARCHAR(255)

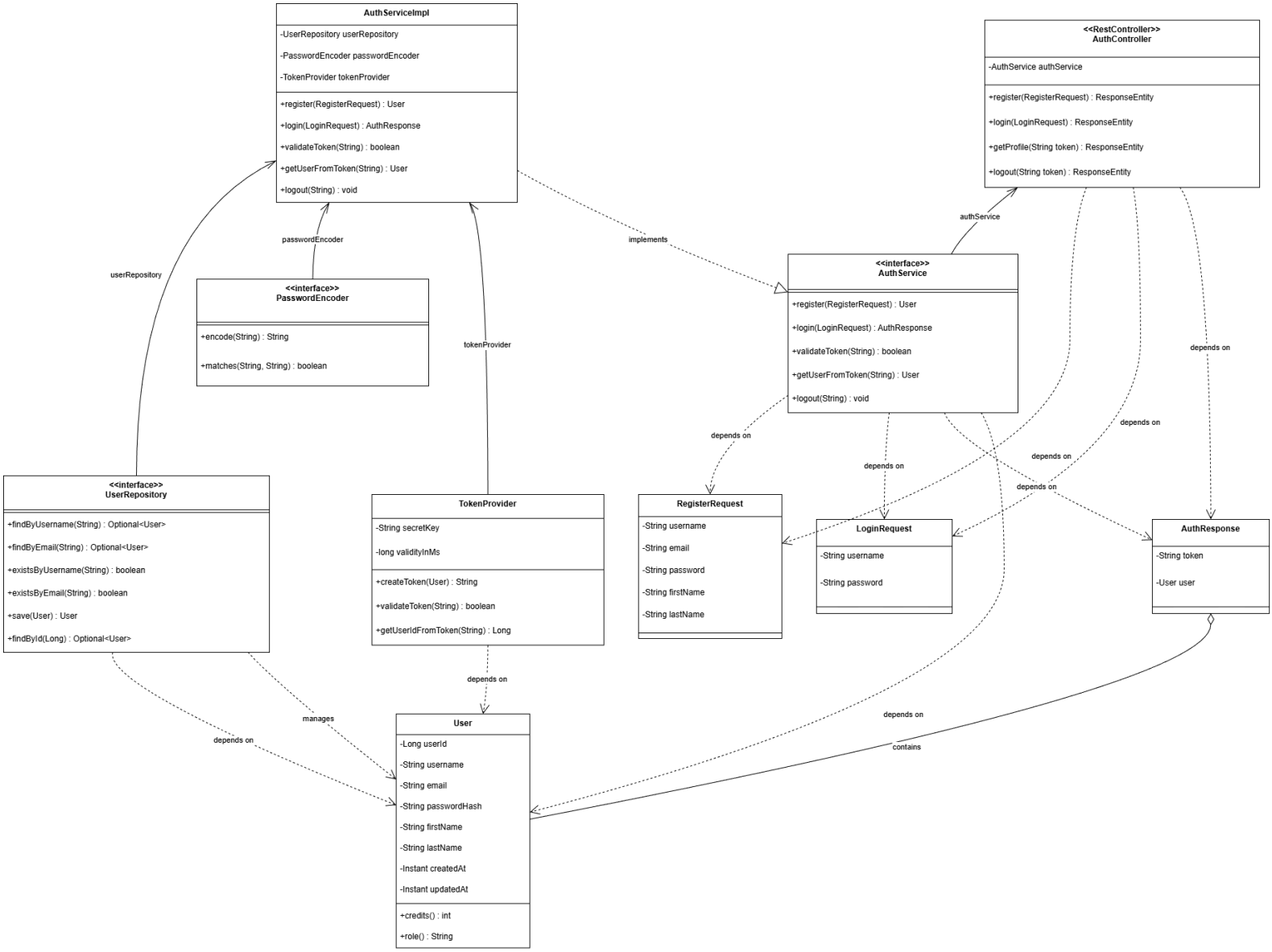
5.2. Use Case Diagram



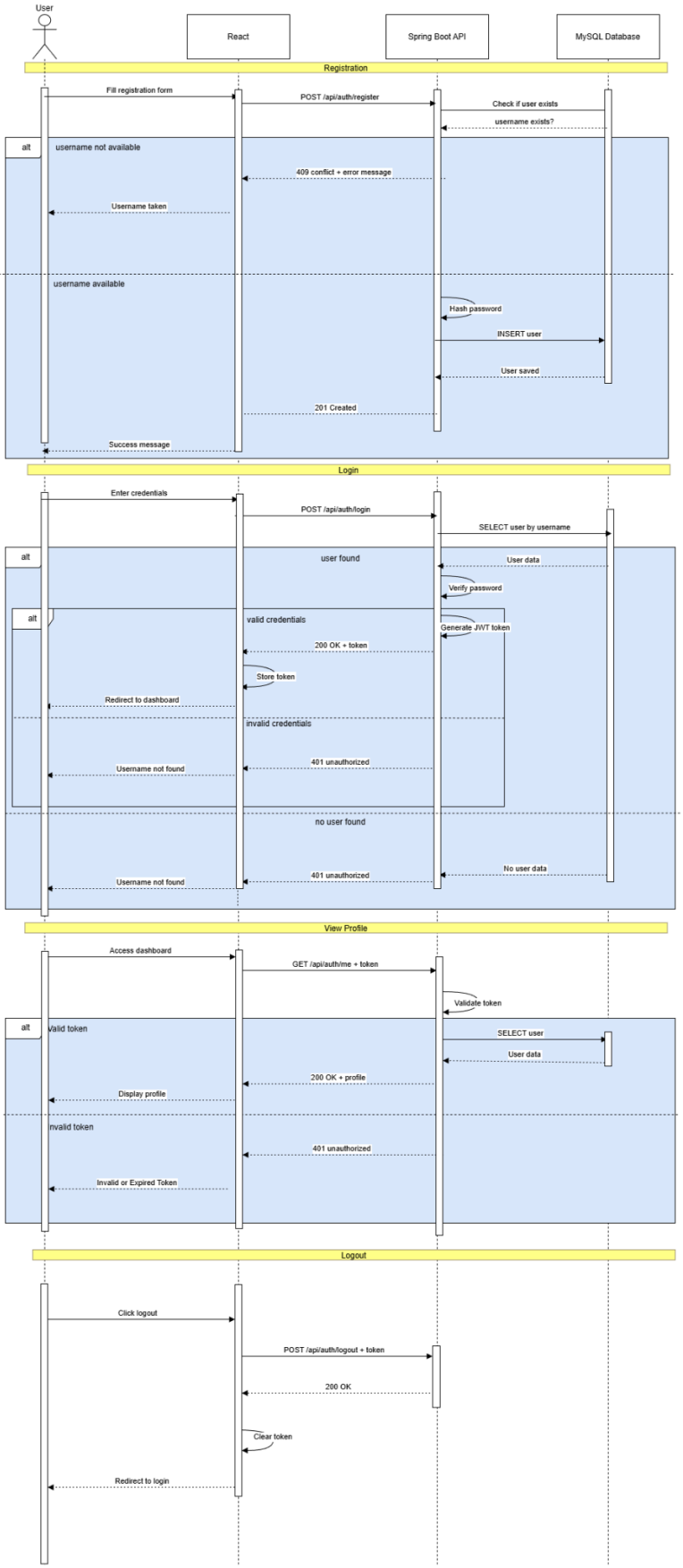
5.3. Activity Diagram



5.4. Class Diagram



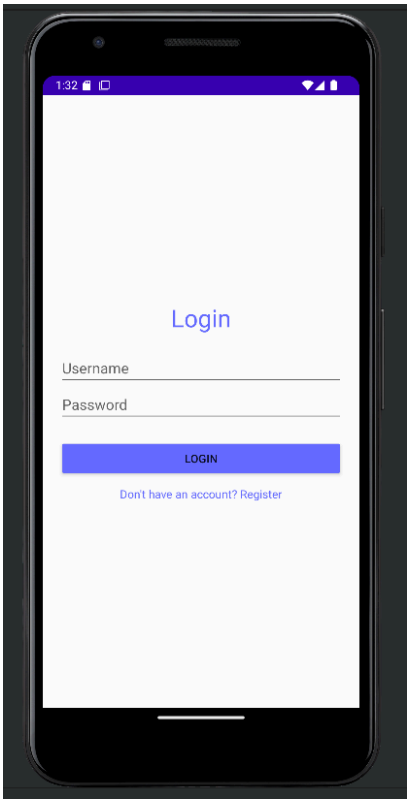
5.5. Sequence Diagram

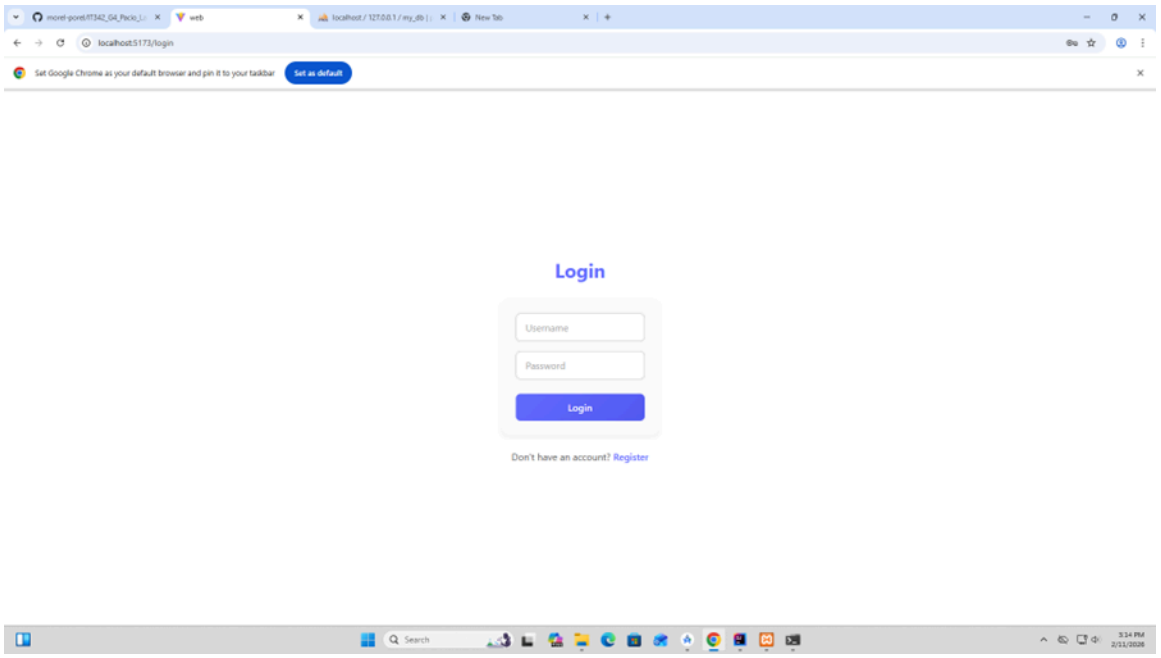


6. Appendices

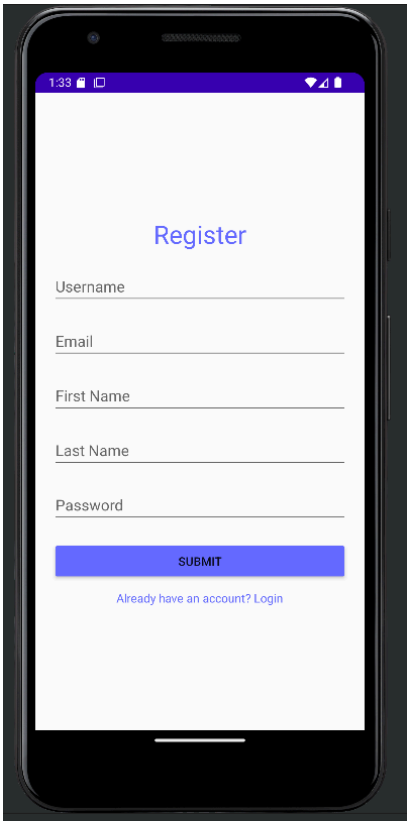
6.1. Final UI Screenshots (web and mobile)

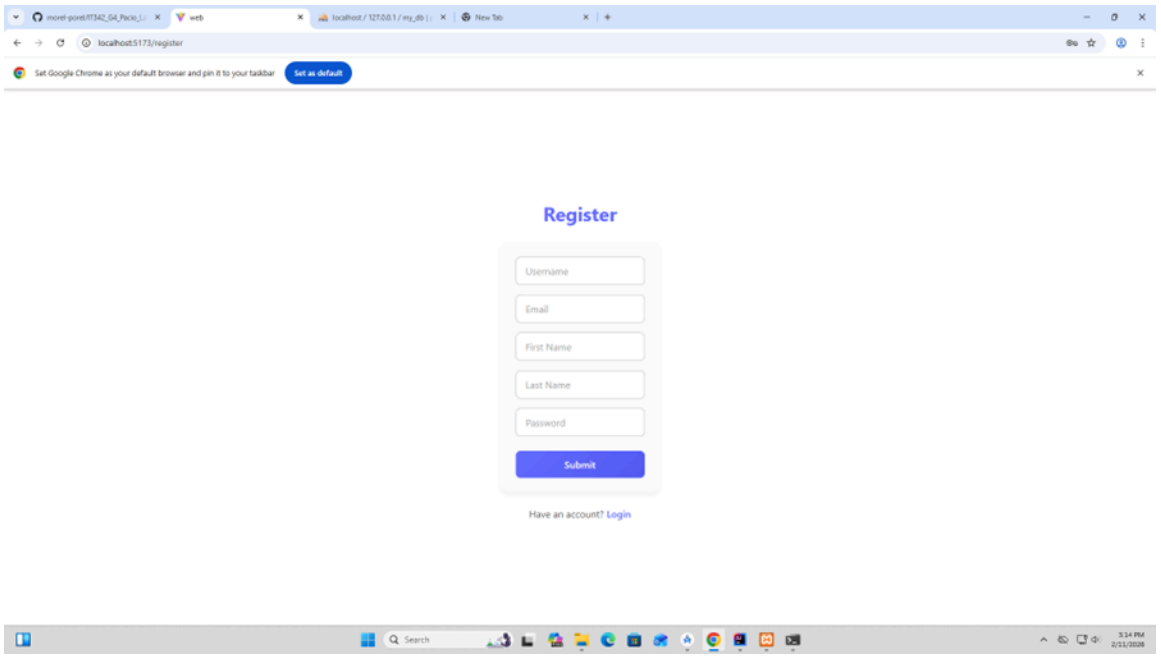
6.1.1.LOGIN SCREEN



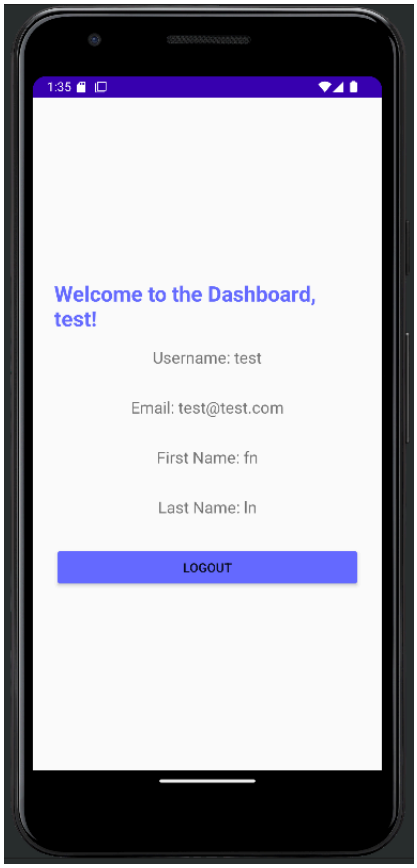


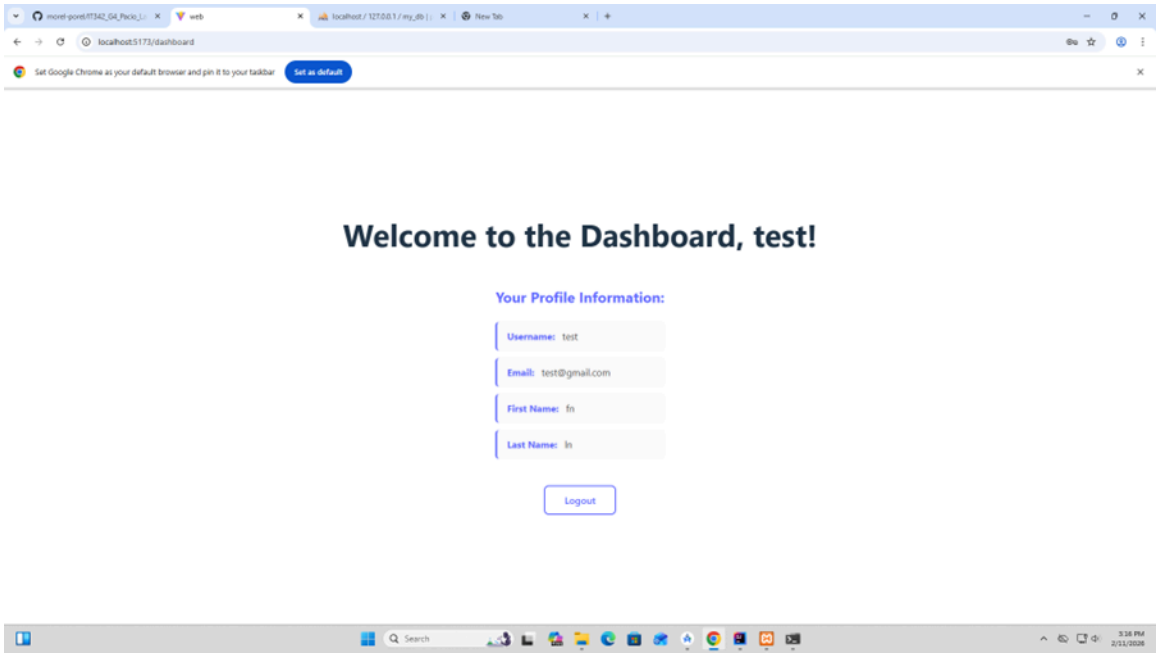
6.1.2.REGISTER SCREEN



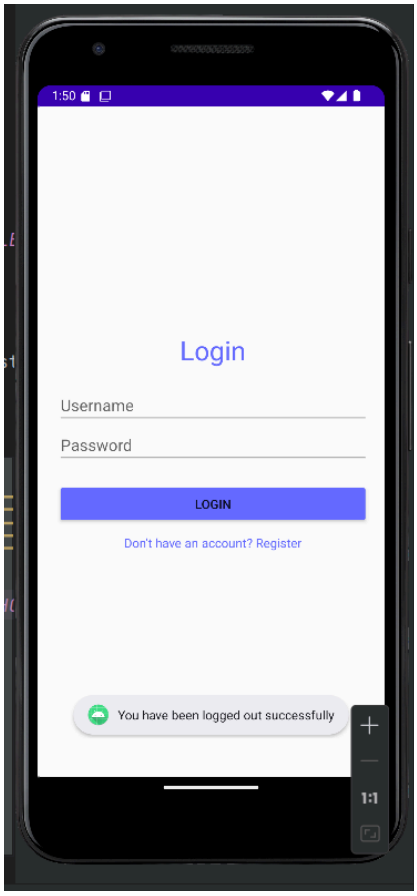


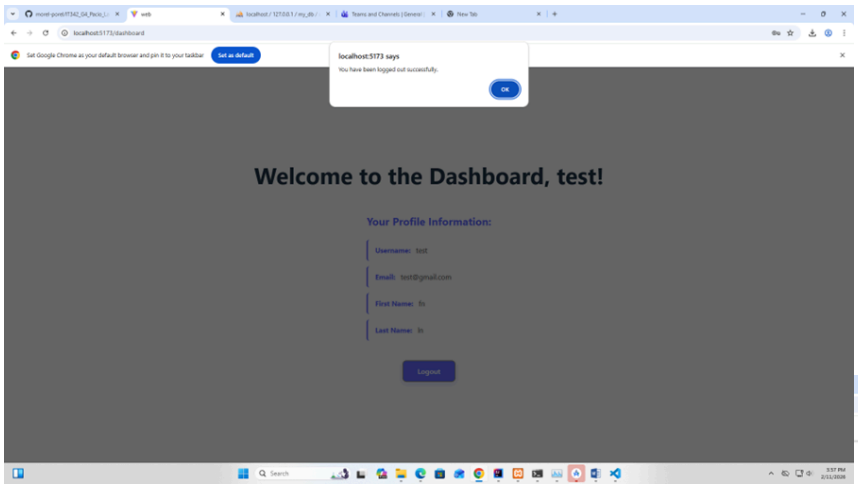
6.1.3.DASHBOARD SCREEN





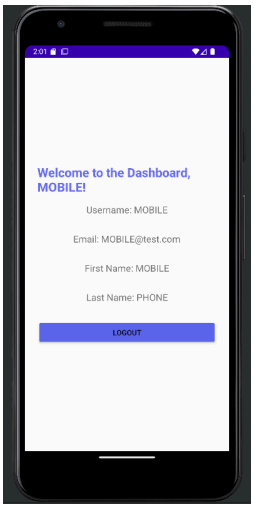
6.1.4.LOGOUT RESULT

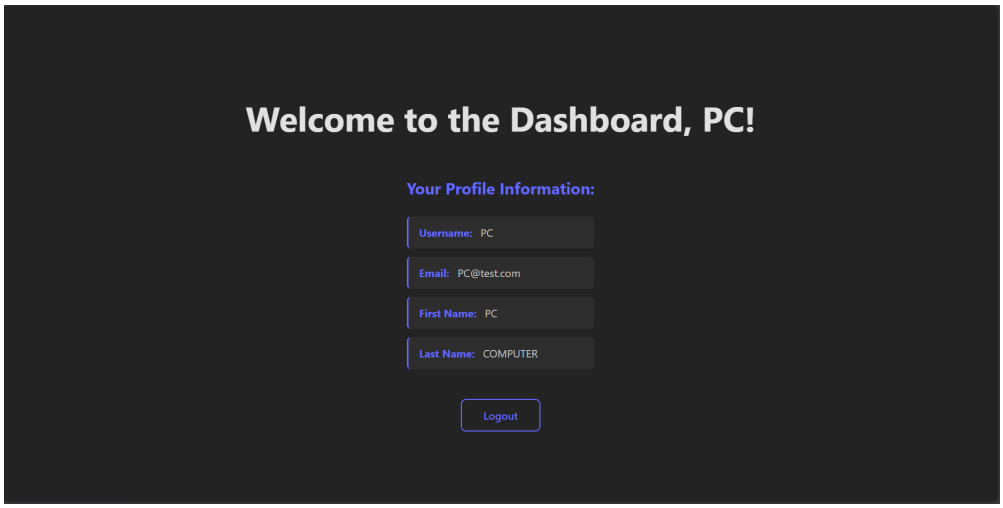




6.2 Proof of Integration

- /me working on both platforms



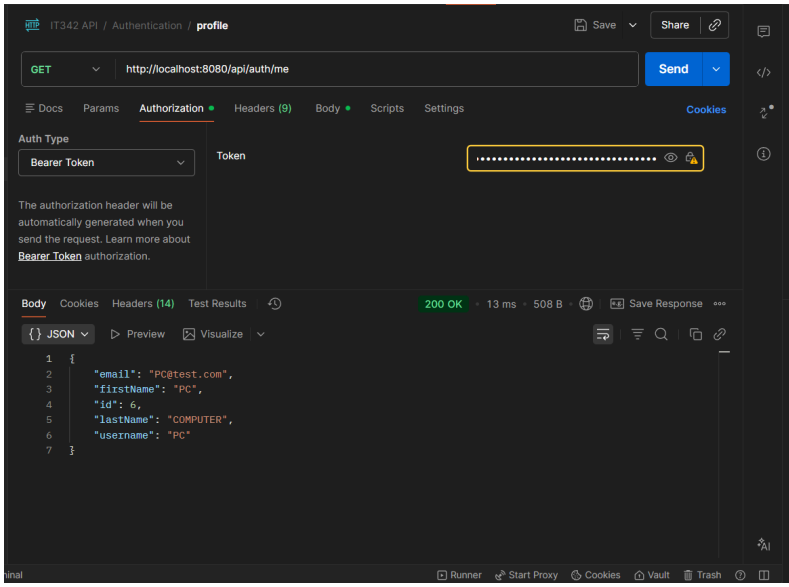


- Backend logs showing the calls from both platforms

```
Hibernate: insert into users (email,first_name,last_name,password_hash,username) values (?,?,?,?,?)
Hibernate: select u1_0.id,u1_0.email,u1_0.first_name,u1_0.last_name,u1_0.password_hash,u1_0.username from users u1_0 where u1_0.username=?
Hibernate: select u1_0.id,u1_0.email,u1_0.first_name,u1_0.last_name,u1_0.password_hash,u1_0.username from users u1_0 where u1_0.id=?
Hibernate: select u1_0.id,u1_0.email,u1_0.first_name,u1_0.last_name,u1_0.password_hash,u1_0.username from users u1_0 where u1_0.id=?
Hibernate: select u1_0.id from users u1_0 where u1_0.username=? limit ?
Hibernate: select u1_0.id from users u1_0 where u1_0.email=? limit ?
Hibernate: insert into users (email,first_name,last_name,password_hash,username) values (?,?,?,?,?)
Hibernate: select u1_0.id,u1_0.email,u1_0.first_name,u1_0.last_name,u1_0.password_hash,u1_0.username from users u1_0 where u1_0.username=?
Hibernate: select u1_0.id,u1_0.email,u1_0.first_name,u1_0.last_name,u1_0.password_hash,u1_0.username from users u1_0 where u1_0.id=?
```

- Postman test showing the same endpoints used by both

PC



MOBILE