



**CEBU INSTITUTE OF TECHNOLOGY**  
**U N I V E R S I T Y**

# IT342-G3

## SYSTEMS INTEGRATION AND ARCHITECTURE 1

---

### **FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)**

---

Project Title: User Registration & Authentication System

Prepared By: Michaela Ma. Alexa D. Estrera

Date of Submission: 2/6/26

Version: 2

# Table of Contents

1.	Introduction	3
1.1.	Purpose	3
1.2.	Scope	3
1.3.	Definitions, Acronyms, and Abbreviations	3
2.	Overall Description	3
2.1.	System Perspective	3
2.2.	User Classes and Characteristics	3
2.3.	Operating Environment	3
2.4.	Assumptions and Dependencies	3
3.	System Features and Functional Requirements	3
3.1.	Feature 1:	3
3.2.	Feature 2:	3
4.	Non-Functional Requirements	3
5.	System Models (Diagrams)	4
5.1.	ERD	4
5.2.	Use Case Diagram	4
5.3.	Activity Diagram	4
5.4.	Class Diagram	4
5.5.	Sequence Diagram	4
6.	Appendices	4

## 1. Introduction

### 1.1. Purpose

This system provides registration, authentication, profile access, and logout functionality. It may be used by stakeholders and developers to understand the system's features, workflow, and architecture.

### 1.2. Scope

- Registration: Guest users will be able to create new user accounts provided their credentials meet the system requirements.
- Log In: Users who enter valid and existing credentials will be successfully logged in to the system.
- Profile Viewing: Authenticated users are able to view their profile.
- Log Out: Authenticated can access the log out function, clearing their credentials and requiring them to log in again to access the profile page.

### 1.3. Definitions, Acronyms, and Abbreviations

- **Access Token** – A short-lived credential (often JWT) that grants a client access to protected resources. Typically included in the Authorization header (e.g., Bearer <token>).
- **Hashing** – A one-way cryptographic function that converts data (e.g., a password) into a fixed-size string. The original input cannot be derived from the hash, ensuring secure password storage and data integrity.
- **JWT (JSON Web Token)** – A compact, URL-safe token (RFC 7519) for securely transmitting claims as a JSON object. It has three Base64Url-encoded parts: Header (algorithm/type), Payload (claims like user ID or roles), and Signature (for verification). Commonly used as Access or ID tokens.
- **Refresh Token** – A long-lived credential issued with an Access Token to obtain a new Access Token without re-entering credentials. Stored securely (e.g., HTTP-only cookie) and sent only to a dedicated refresh endpoint.

## 2. Overall Description

### 2.1. System Perspective

This system is a modular service that manages user registration, authentication, profile viewing, and logout functionality. It interacts with a frontend client (React UI) via a RESTful API built in Spring Boot and persists user data in a backend database. The system handles input validation, secure password storage, session management using JWT access and refresh tokens, and enforces access control for protected resources. Its modular design allows integration with other services in the future.

### 2.2. User Classes and Characteristics

- Guest User - A person who intends to access the system's functions but does not have a user account yet.
- Authenticated User - A user with verified credentials from the back end. This user can view the profile page and use the log out function.

### 2.3. Operating Environment

- Modern web browsers (Chrome, Firefox, Edge, Safari) with JavaScript support
- Reliable internet connection
- Devices capable of running web applications (desktop, laptop, tablet)
- Backend server running Spring Boot connected to a database

### 2.4. Assumptions and Dependencies

#### Assumptions

- Users have stable internet access and compatible browsers
- Backend and database are properly configured and accessible
- Users follow standard security practices (e.g., not sharing credentials)

---

#### Dependencies

- Server uptime and availability for handling requests
- Database availability for storing and retrieving user data
- Authentication service (JWT generation and validation) functioning correctly
- Network infrastructure supports secure client-server communication
- Required frontend and backend frameworks/libraries are maintained

## 3. System Features and Functional Requirements

### 3.1. Feature 1: User Authentication

#### Description:

Handles user registration and login to grant access to protected resources.

#### Functional Requirements:

- Users can register by providing a unique email, username, and password.
- Users can log in using valid credentials; invalid credentials return an error.
- The system securely stores passwords (hashed) and issues JWT tokens upon login.

---

### 3.2. Feature 2: Profile Page

#### Description:

Allows authenticated users to view their profile.

#### Functional Requirements:

- Authenticated users can view their profile details (email, username).
- Access to profile pages is restricted to authenticated users; guests attempting access receive an error (403).

### 3.3. Feature 3: Log Out

**Description:**

Allows authenticated users to log out.

**Functional Requirements:**

- The system removes the JWT refresh token associated with the user from the database.
- The user credentials are cleared from the front end, and they are prompted to log in again to access the profile page.

## 4. Non-Functional Requirements

### Performance

- The system shall respond to user registration, login, profile viewing, and logout requests within **2 seconds** under normal operating conditions.

### Security

- User passwords shall be **hashed and salted** before being stored in the database.
- Authentication shall use **JWT tokens**.
- Access to protected resources, such as the profile page, shall be restricted to **authenticated users only**.

### Usability

- The user interface shall be intuitive and easy to navigate.
- Clear and user-friendly error messages shall be displayed for invalid credentials and duplicate registrations.

### Reliability

- The system shall maintain at least **99% uptime**, excluding scheduled maintenance.

## Maintainability

- The system shall use a **modular code structure** to allow easy updates and future feature additions such as password recovery or third-party authentication.

## Compatibility

- The system shall function correctly on the latest versions of **Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari**.

## Scalability

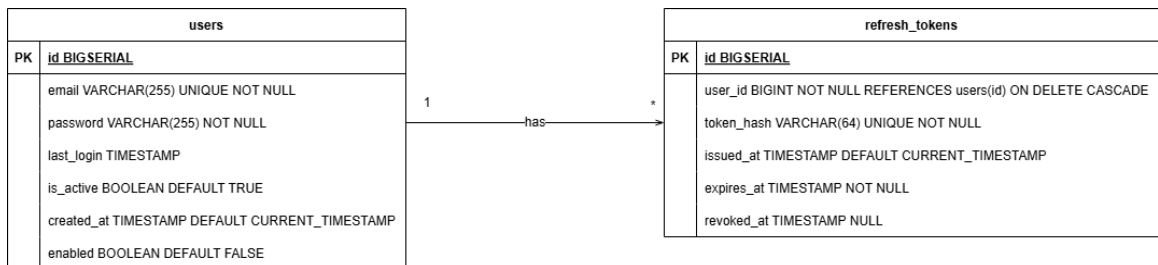
- The system shall support an increasing number of users without significant performance degradation through scalable backend design.

## Availability

- The system shall be available **24/7**, except during scheduled maintenance periods which shall be announced in advance.

## 5. System Models (Diagrams)

### 5.1. ERD

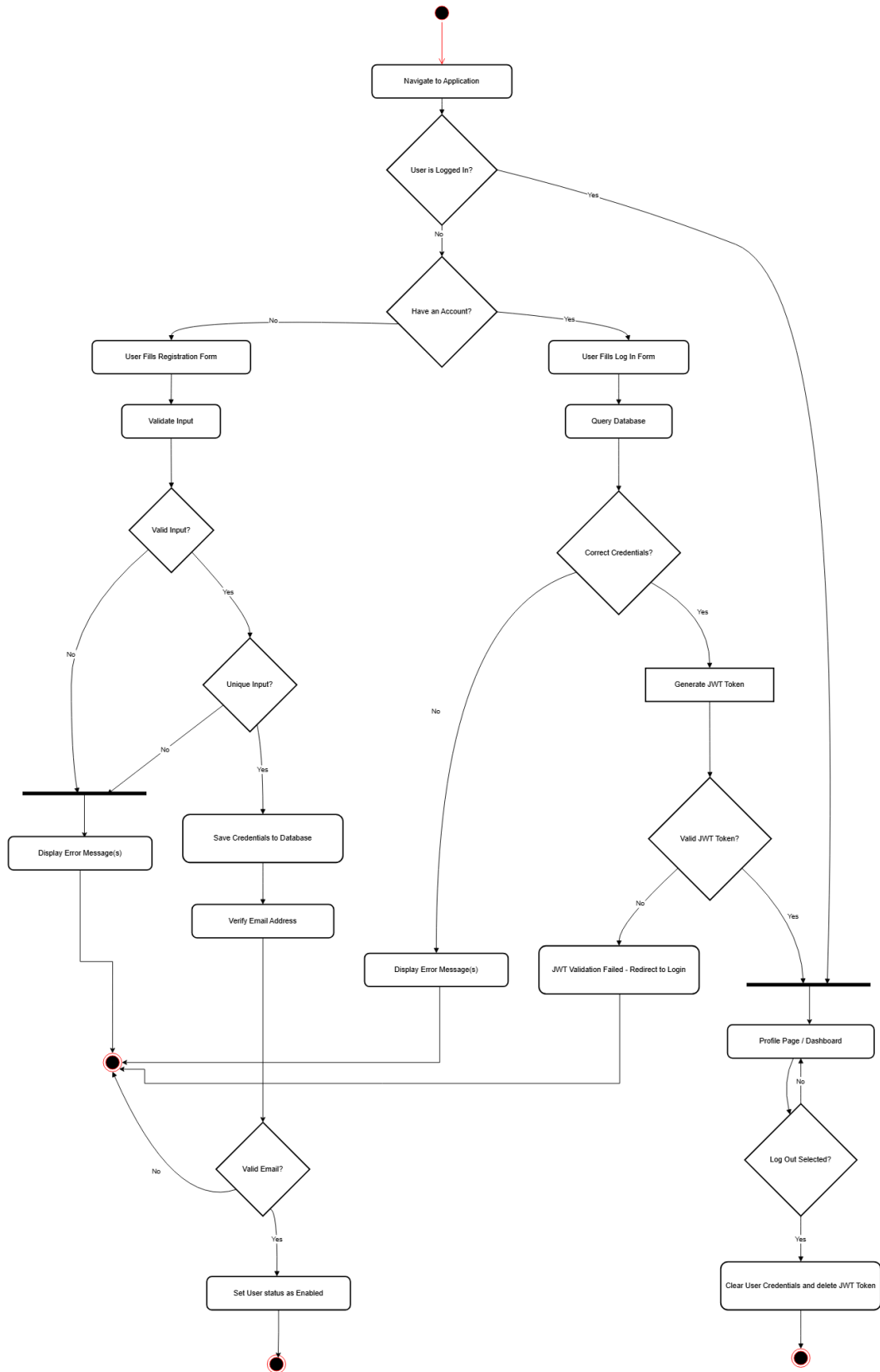


## 5.2. Use Case Diagram

### User Registration & Authentication Use Case Diagram

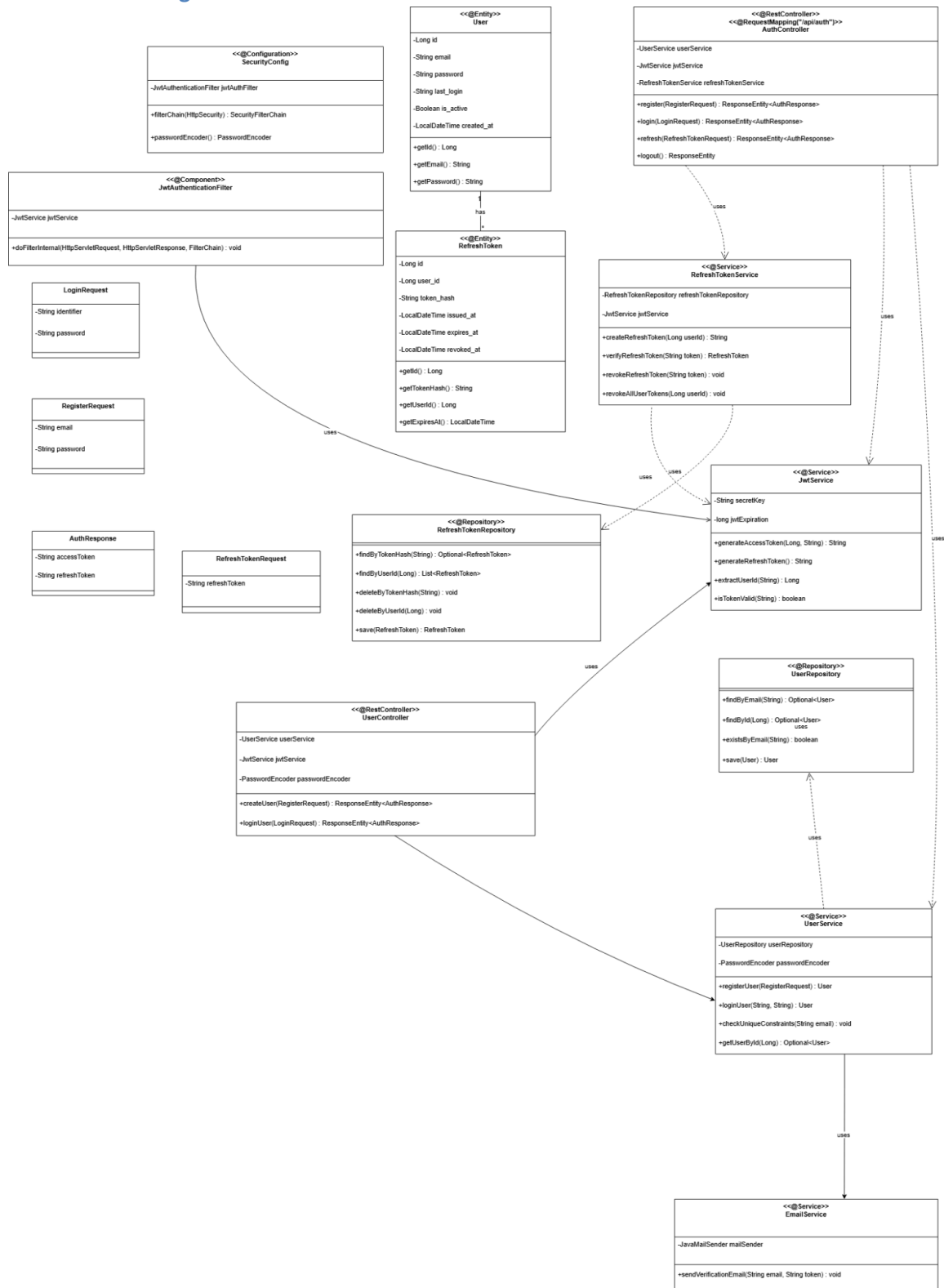


### 5.3. Activity Diagram



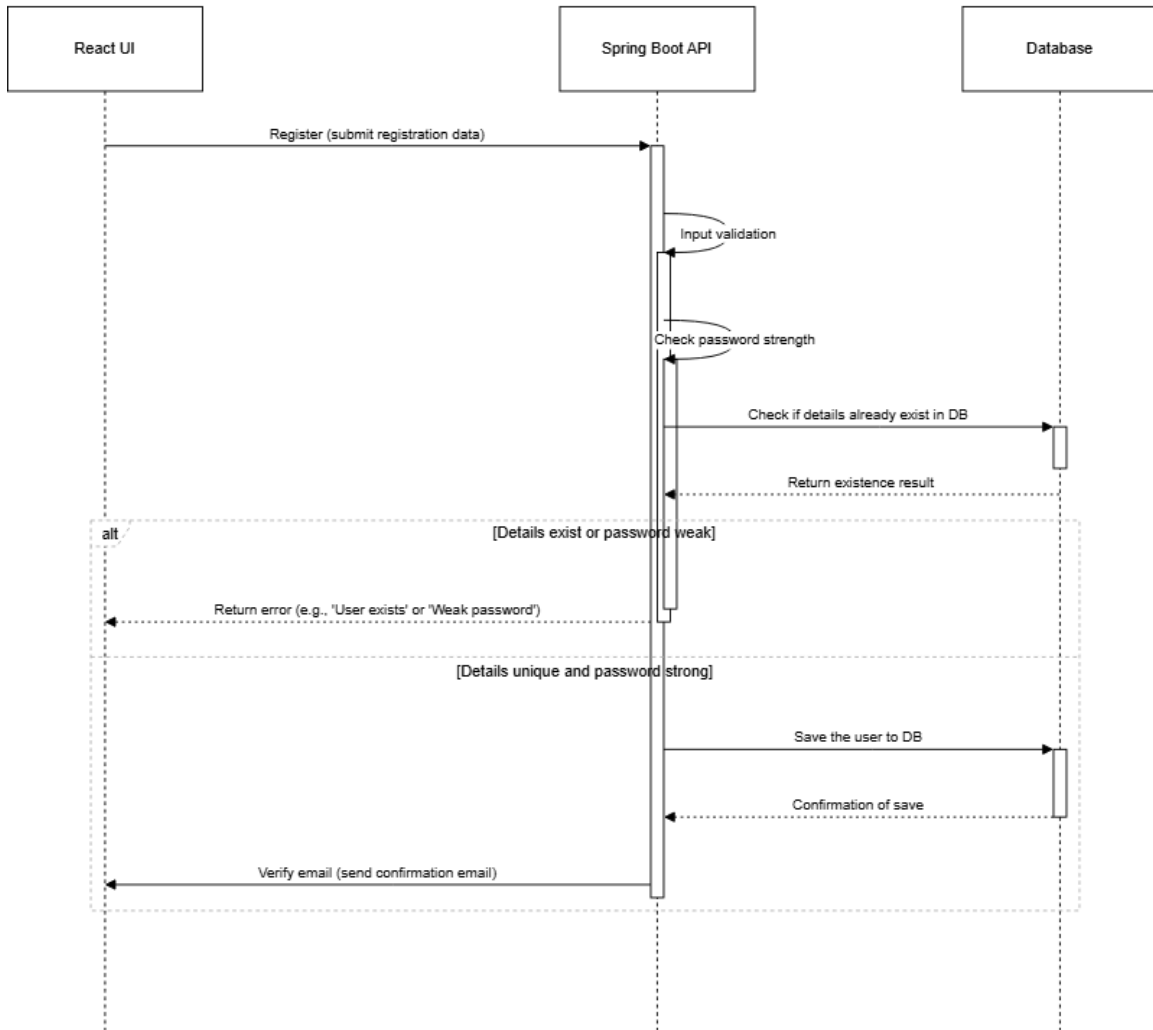


## 5.4. Class Diagram

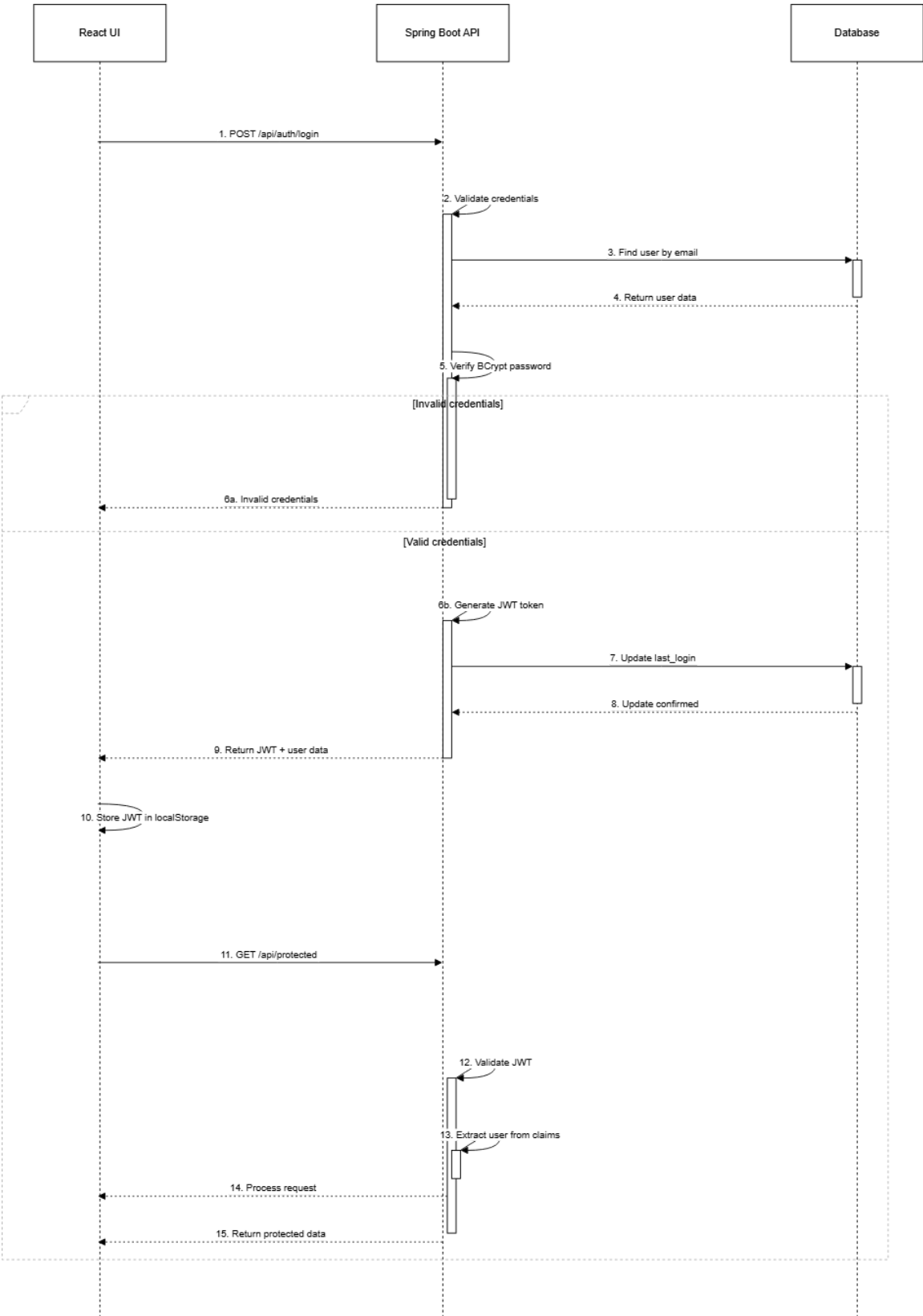


## 5.5. Sequence Diagram

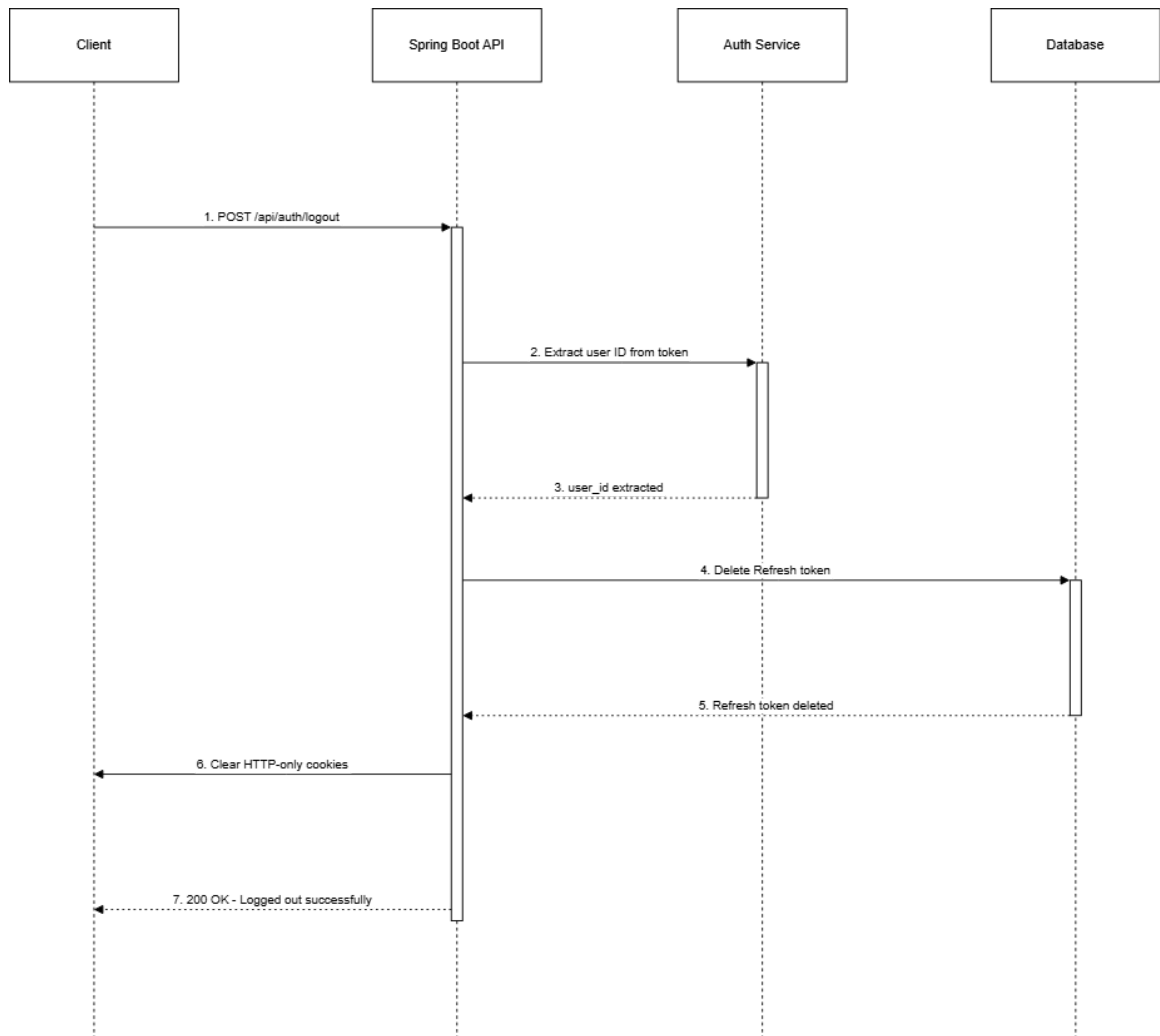
### Registration Sequence Diagram



# Log In Sequence Diagram



## Log Out Sequence Diagram



## 6. Appendices

[1] "Use Case Diagram Tutorial." Accessed: Feb. 06, 2026. [Online]. Available: <https://online.visual-paradigm.com/diagrams/tutorials/use-case-diagram-tutorial/>