



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

# IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

---

## FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

---

Project Title: User Registration and Authentication Prepared

By: Dabon, Kenn Xavier C.

Date of Submission: 2/6/26

Version: 1.0

# 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

The purpose of this system is to provide a secure and scalable foundation for user identity management and authorization. It allows users to create accounts and authenticate securely. This document is intended for software developers, system architects, and testers to understand the system's design and functional expectations.

### 1.2. Scope

The system will facilitate:

- Secure user registration with data validation.
- Token-based authentication (JWT) for login.
- Protected access to user dashboards and profile editing.
- Session termination (Logout).

### 1.3. Definitions, Acronyms, and Abbreviations

- JWT: JSON Web Token
- DTO: Data Transfer Object
- API: Application Programming Interface
- UI: User Interface (ReactJS)

## 2. Overall Description

### 2.1. System Perspective

The system is a basic authentication application that links a React-based frontend with a Spring Boot backend and a database. It operates within a larger software environment, allowing users to interact through a web and mobile interface while the backend handles authentication logic and data storage.

### 2.2. User Classes and Characteristics

This system has two different types of users:

- Guest: A user who has not logged in. Can ONLY access the Login and Register pages.
- Registered User: A user who has successfully created an account. They can log in, view the dashboard, and edit their profile details.

### 2.3. Operating Environment

Client: Any modern web browser (Chrome, Firefox, Edge, Safari, etc.) with JavaScript enabled.

Server: Spring Boot

Database: MySQL

### 2.4. Assumptions and Dependencies

Users must have a stable internet connection.

Passwords must be stored as hashes, never as plain text.

## 3. System Features and Functional Requirements

### 3.1. Feature 1:

Description: Allows a guest to create a new account by providing the necessary credentials.

Functional Requirements:

- The system shall validate the email provided is in a correct format.

- The system shall check if the email or username already exists in the database.
- The system shall hash the user's password before saving to the database.
- The system shall create a User entity upon successful validation.

### 3.2. Feature 2:

Description: Allows a registered user to gain access to the system by verifying their identity.

Functional Requirements:

- The system shall accept a LoginDto containing username and password.
- The system shall verify the provided password against the stored password hash.
- If credentials are valid, the system shall generate a signed JWT token. Otherwise, the system returns an Unauthorized error message.

### 3.3. Feature 3:

Description: Allows a logged-in users to view and update their personal information

Functional Requirements:

- The system shall require a valid JWT token in the HTTP Authorization header to access profile data.
- The system shall allow users to view their username and email.
- The system shall allow users to update their profile details and save change.

### 3.4. Feature 4:

Description: Securely terminates the user's session

Functional Requirements:

- The Client shall remove the JWT token from local storage upon clicking "Logout".
- The Client shall redirect the user to the Login page immediately after logging out.
- The system shall prohibit access to protected routes (Dashboard) once the token is removed.

## 4. Non-Functional Requirements

### Security

- All passwords must be encrypted.
- JWT tokens must expire after a set duration (e.g., 24 hours).

### Performance

- Login and Registration requests should be processed in under **200ms**.
- The system should support concurrent login requests without degradation.

## Reliability

- The system should prevent data corruption in case of a failed transaction.

## Usability

- The UI must provide clear error messages (e.g., "Incorrect Password", "Email already in use") to the user.

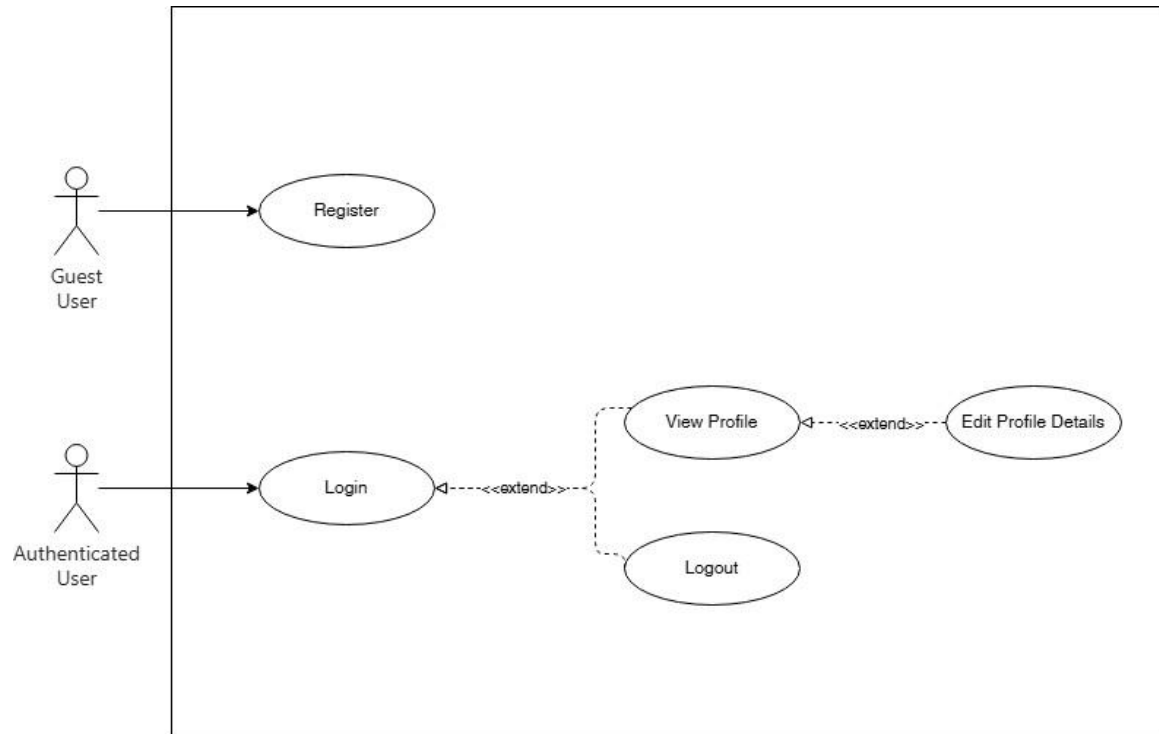
## 5. System Models (Diagrams)

### 5.1. ERD

User		
PK	<u>user_id</u>	<u>INTEGER</u>
	username	VARCHAR(255)
	email	VARCHAR(255)
	password	VARCHAR(255)
	first_name	VARCHAR(255)
	last_name	VARCHAR(255)
	created_at	TIMESTAMP

## 5.2.

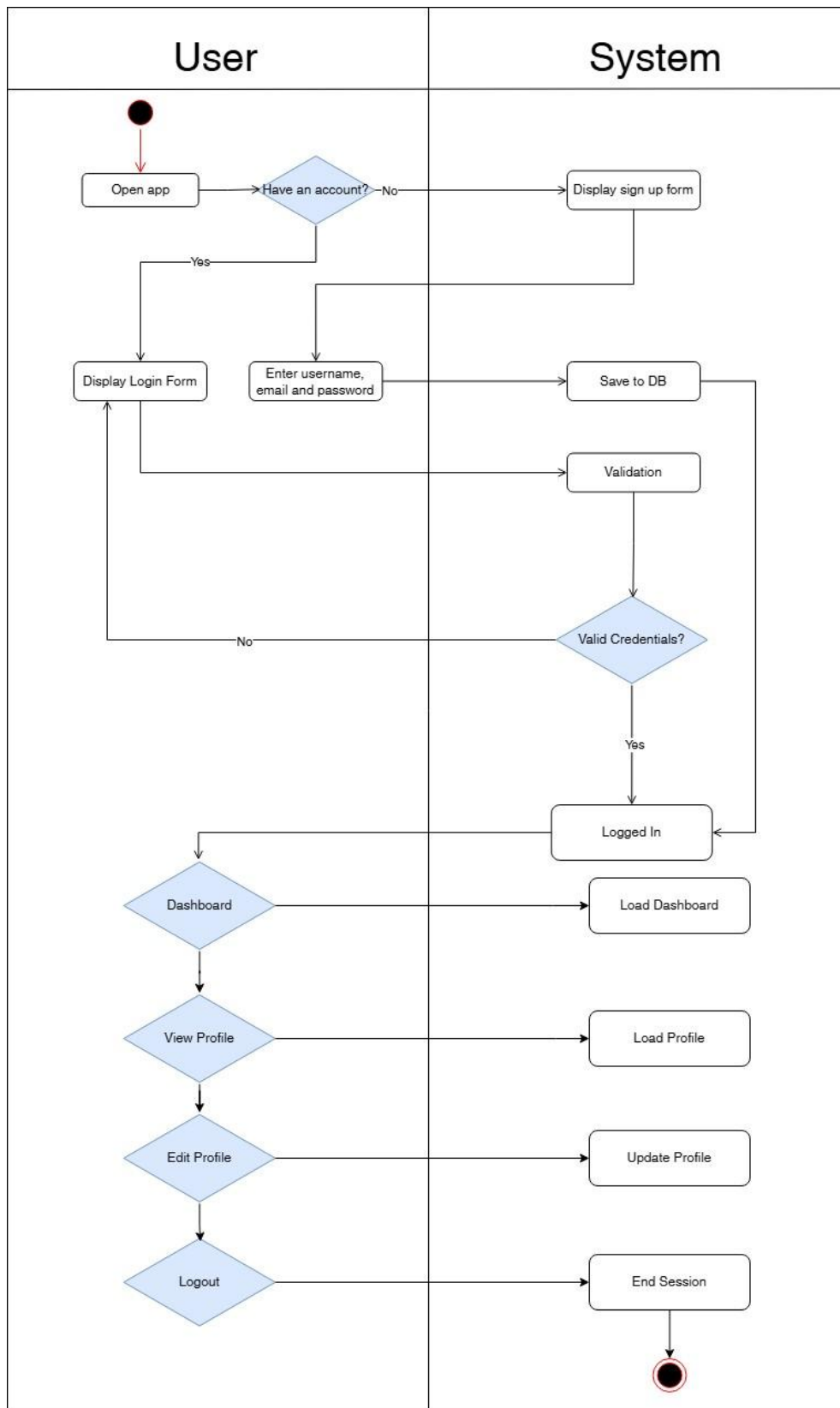
### Use Case Diagram



### 5.3.

#### Activity Diagram

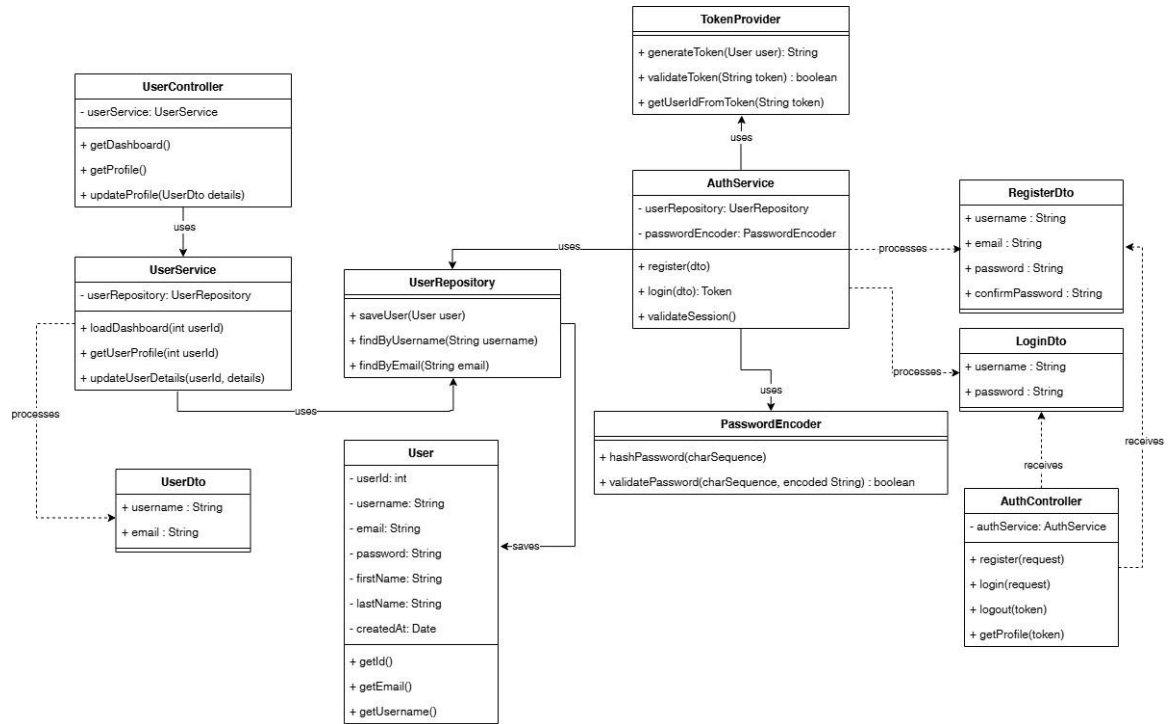
#### 5.4.



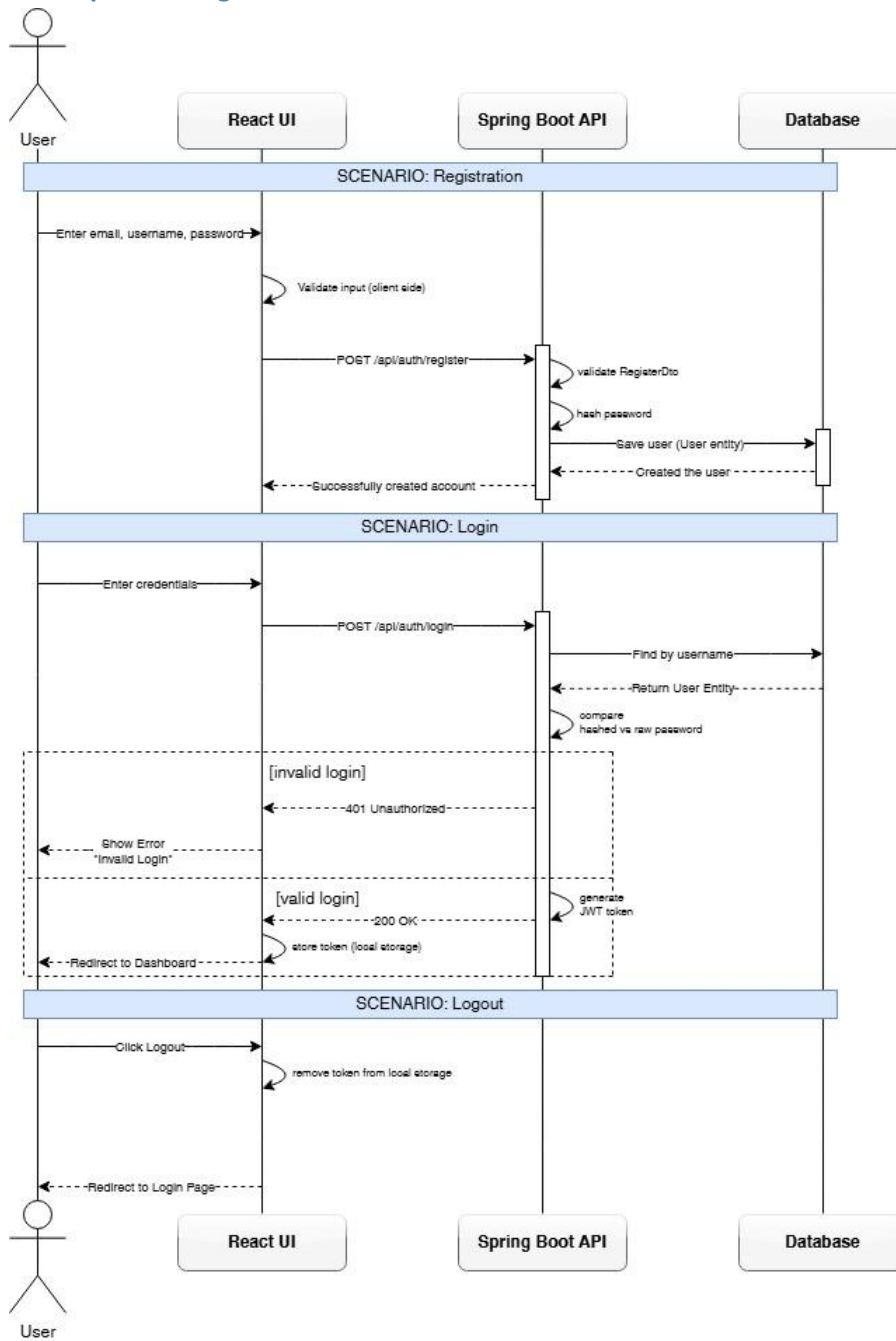


## 5.5.

### Class Diagram

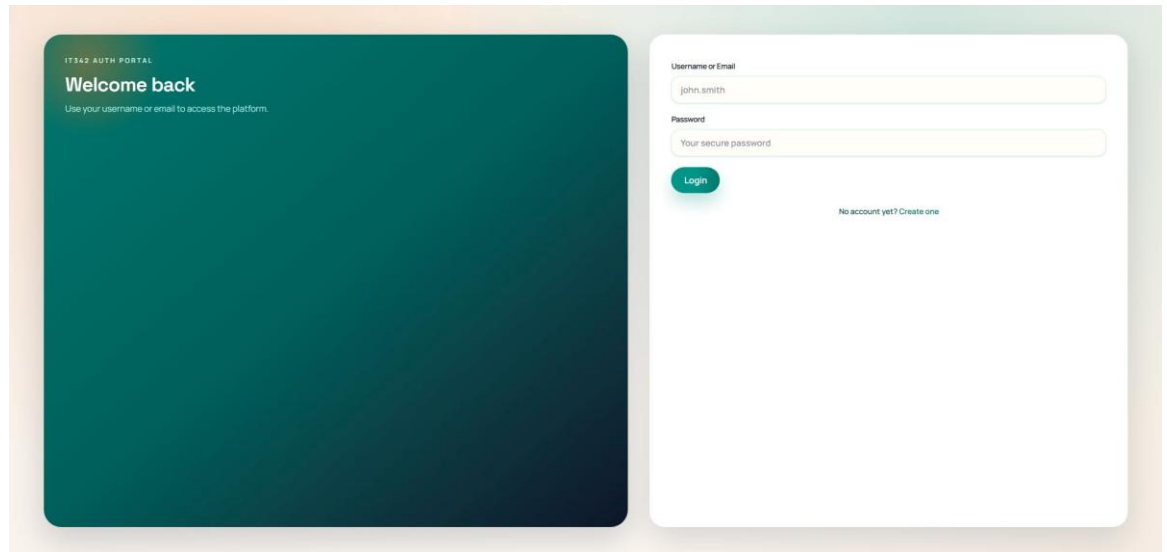


## 5.6. Sequence Diagram



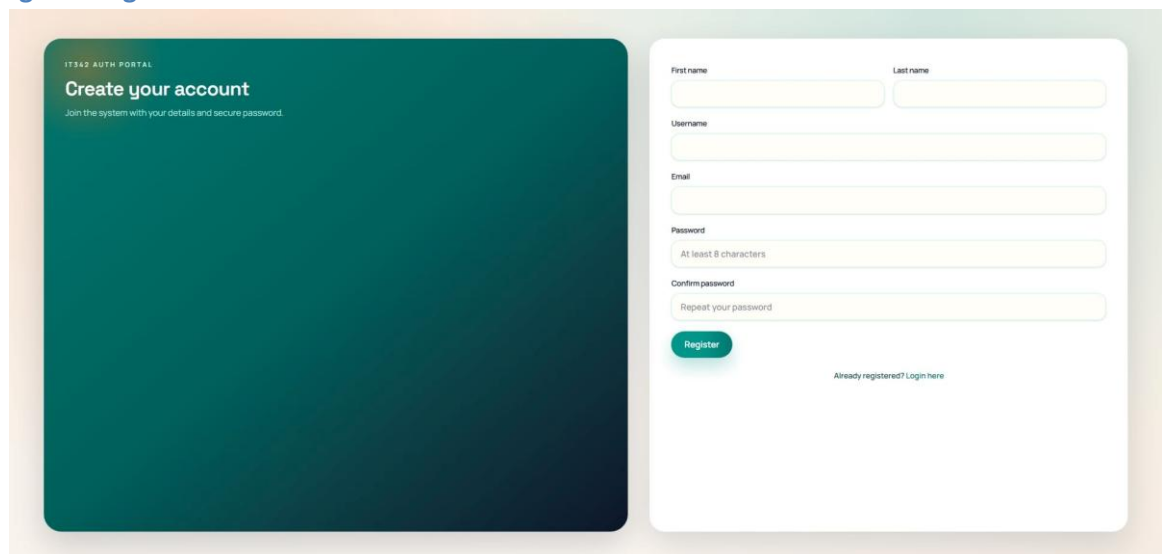
## 6. ReactJS Web UI

### 6.1. Login Page



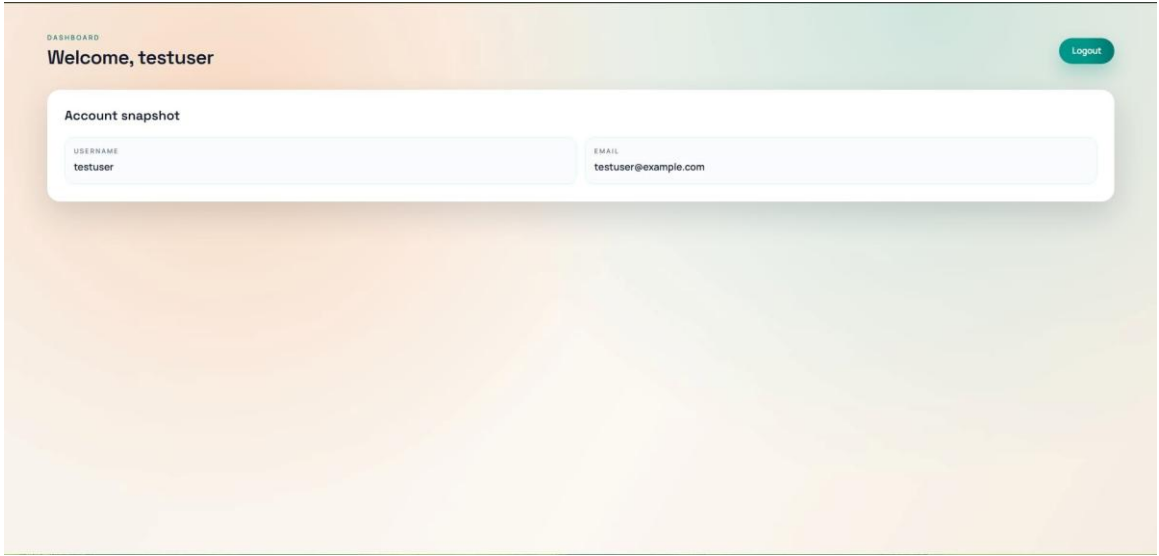
The login page features a dark teal sidebar on the left with the text "IT362 AUTH PORTAL" and "Welcome back" followed by the instruction "Use your username or email to access the platform." The main content area is white and contains a login form with fields for "Username or Email" (containing "john.smith") and "Password" (containing "Your secure password"). A green "Login" button is positioned below the password field. At the bottom right, there is a link that says "No account yet? Create one".

### 6.2. Register Page



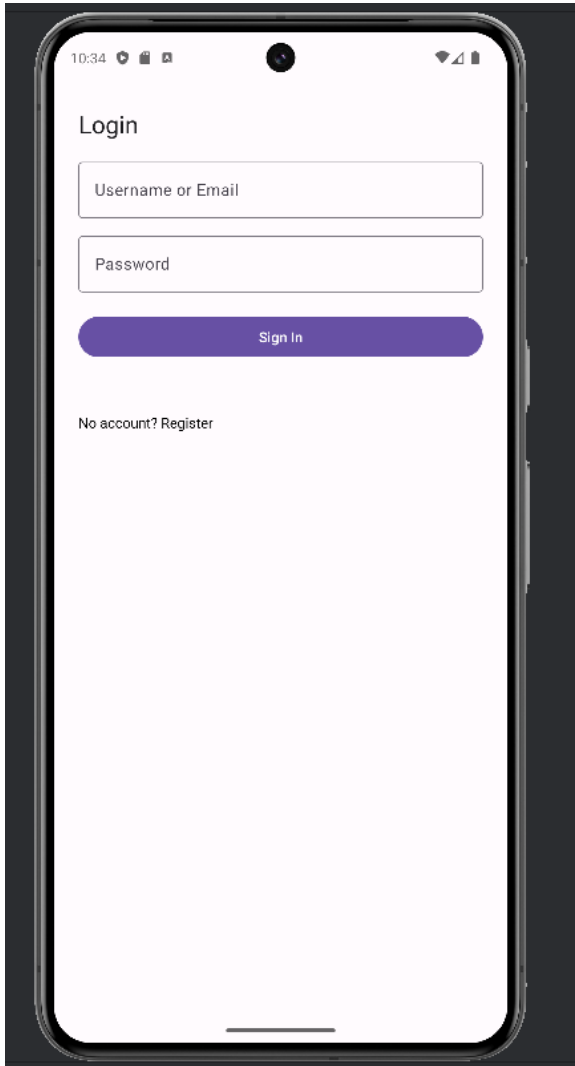
The register page features a dark teal sidebar on the left with the text "IT362 AUTH PORTAL" and "Create your account" followed by the instruction "Join the system with your details and secure password." The main content area is white and contains a registration form with fields for "First name", "Last name", "Username", "Email", "Password" (with a note "At least 8 characters"), and "Confirm password" (with a note "Repeat your password"). A green "Register" button is positioned below the confirm password field. At the bottom right, there is a link that says "Already registered? Login here".

### 6.3. Dashboard



## 6. Kotlin Mobile UI

### 6.1. Login Page



6.2. Register Page

The image shows a mobile application interface for a registration page. The screen is titled "Register" and features a light pink background. At the top, there is a status bar with the time "10:34" and various icons. Below the title, there are five input fields stacked vertically: "Username", "Email", "Password", "First Name", and "Last Name". Each field has a light purple border and a small icon on the right side. Below these fields is a prominent purple button labeled "Create Account". At the bottom of the form, there is a link that says "Have an account? Sign In". The entire form is enclosed in a dark grey border, which is part of a mobile phone frame.

10:34

## Register

Username

Email

Password

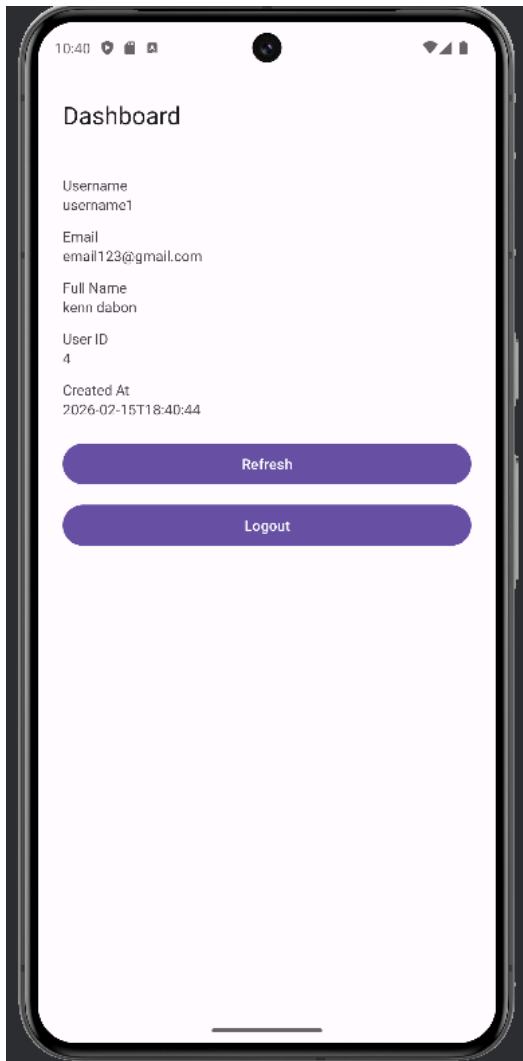
First Name

Last Name

Create Account

Have an account? Sign In

## 6.2. Dashboard



## 7. Appendices

Diagrams created using <https://www.drawio.com/>

Additional guides:

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/> <https://venngage.com/blog/class-diagram/>