Contents

[1. Overview 2](#_Toc154426618)

[2. Project Structure 3](#_Toc154426619)

[3. Functionality Overview 4](#_Toc154426620)

[3.1 Care Giver Functionality 4](#_Toc154426621)

[3.2 Chat Functionality 4](#_Toc154426622)

[3.3 Recipient Functionality 4](#_Toc154426623)

[3.4 Team Functionality 4](#_Toc154426624)

[3.5 User Login and Registration 4](#_Toc154426625)

[4. Roles and Operations 5](#_Toc154426626)

[4.1 Admin Role 5](#_Toc154426627)

[4.2 Care Giver Role 5](#_Toc154426628)

[4.3 Customer Role 5](#_Toc154426629)

[4.4 Team Role 6](#_Toc154426630)

[5. Technologies Used 6](#_Toc154426631)

[6. Getting Started 6](#_Toc154426632)

[2. Email Configuration: 7](#_Toc154426633)

[3. Multipart File Configuration: 7](#_Toc154426634)

[7. Installation 8](#_Toc154426635)

[7.1 Prerequisites 8](#_Toc154426636)

[7.2 Installation 8](#_Toc154426637)

[7. Development 9](#_Toc154426638)

[7.1 Care Giver Controller 9](#_Toc154426639)

[7.2 Chat Controller 9](#_Toc154426640)

[7.3 Recipient Controller 9](#_Toc154426641)

[7.4 Team Controller 9](#_Toc154426642)

[7.5 User Login and Registration Controller 9](#_Toc154426643)

[8. Models in Details 10](#_Toc154426644)

[9. Repositories and Database Operations 12](#_Toc154426645)

# 1. Overview

The Customer Care Registry Spring Boot Project is a comprehensive system designed to facilitate customer care operations, complaint management, and team collaboration within a Spring Boot framework. The project encompasses various modules, each catering to specific roles such as Care Givers, Customers, and Admins.

The application is structured to provide seamless communication channels, complaint registration, real-time chat functionality, and efficient team management. With a robust set of features, the project aims to enhance the customer care experience and streamline internal processes.

This document provides an in-depth overview of the project's structure, functionality, roles and operations, technologies used, and how to get started with development.

Abstract

The Customer Care Registry Spring Boot Project is an integrated system designed to streamline customer care processes, complaint management, and team collaboration within the Spring Boot framework. It comprises distinct modules tailored for Care Givers, Customers, and Admins.

The application is architected to offer smooth communication channels, complaint registration, real-time chat capabilities, and effective team administration. With a comprehensive feature set, the project is geared towards improving the overall customer care experience and optimizing internal workflows.

This abstract provides a detailed insight into the project's organization, functionalities, roles and operations, employed technologies, and guidelines for initiating

# 2. Project Structure

2.1 Controllers

- Contains classes responsible for handling HTTP requests and managing the flow of data between the view and the business logic.

2.2 Service

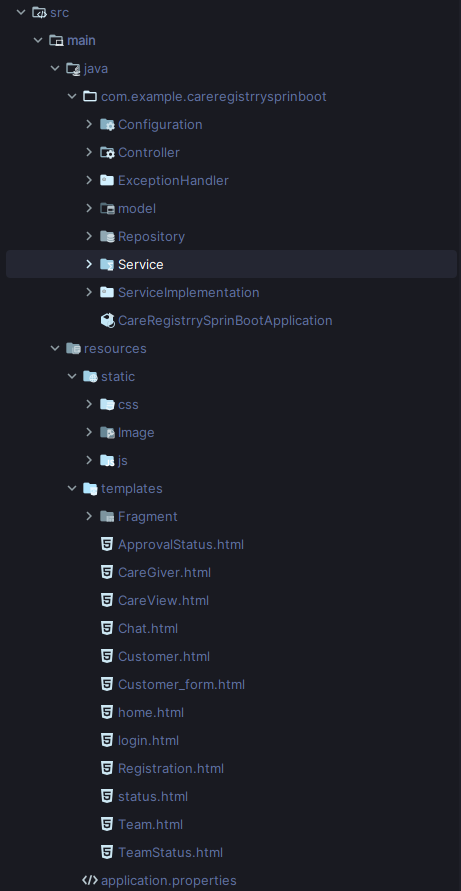
- Houses service classes that contain the business logic of the application. These services are responsible for processing data and interacting with repositories.

2.3 Models

- Includes classes representing the data structure of the application. These classes define the entities used in the system, such as users, care givers, recipients, etc.

2.4 Repositories

- Holds repository interfaces that define methods for interacting with the database. These interfaces are typically extended by Spring Data JPA, providing database access.



# 3. Functionality Overview

## 3.1 Care Giver Functionality

* View and update care giver information.
* View care recipient information.
* Save care recipient form data.
* Participate in team chat.

## 3.2 Chat Functionality

* Send and receive real-time chat messages.
* Join and leave chat sessions.
* Enhance user communication and collaboration.

## 3.3 Recipient Functionality

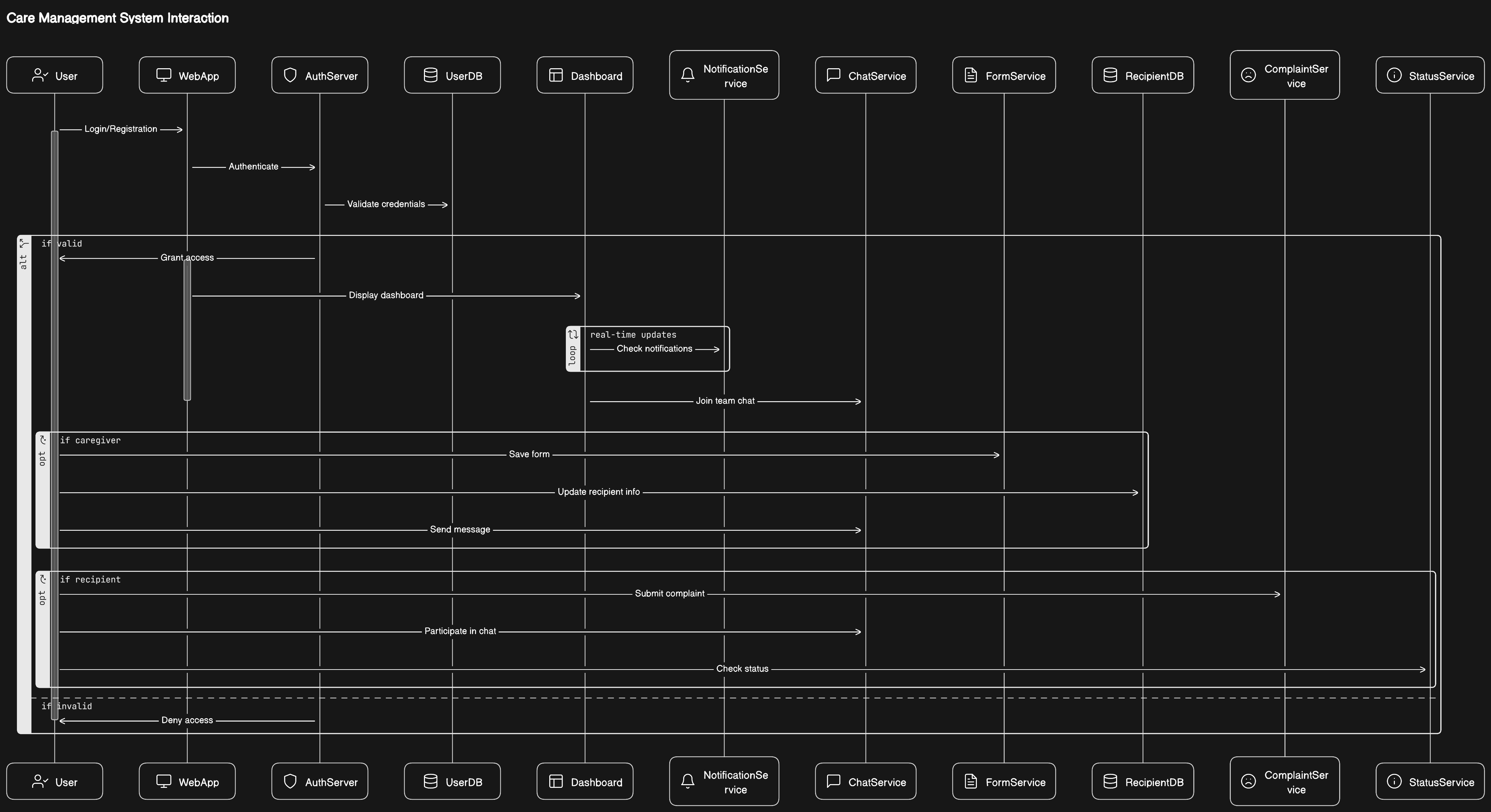
* Submit complaints with associated files.
* Receive email notifications.
* Check complaint status.
* Participate in real-time chat.

## 3.4 Team Functionality

* View team dashboard.
* Participate in team chat.
* Save decisions related to care recipients.

## 3.5 User Login and Registration

* User registration with role selection.
* User authentication and login.
* Role-based access control.
* Manage user profiles and credentials.



# 4. Roles and Operations

## 4.1 Admin Role

* Manage users (add, remove, update).
* Access to all functionalities.
* Admin dashboard for system monitoring.

## 4.2 Care Giver Role

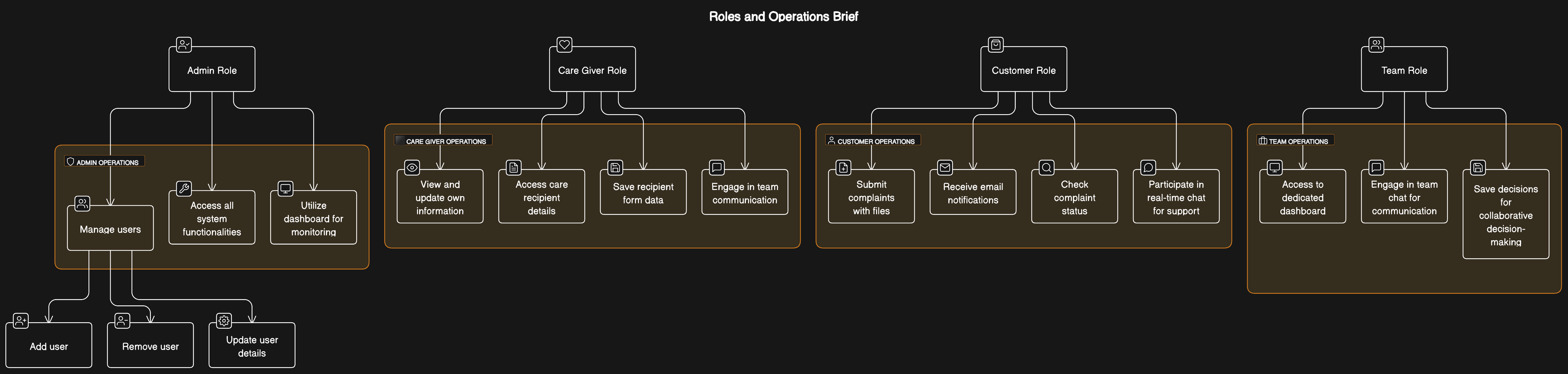
* View and update care giver information.
* View care recipient information.
* Save care recipient form data.
* Participate in team chat.

## 4.3 Customer Role

* Submit complaints with associated files.
* Receive email notifications.
* Check complaint status.
* Participate in real-time chat.

## 4.4 Team Role

* View team dashboard.
* Participate in team chat.
* Save decisions related to care recipients.



# 5. Technologies Used

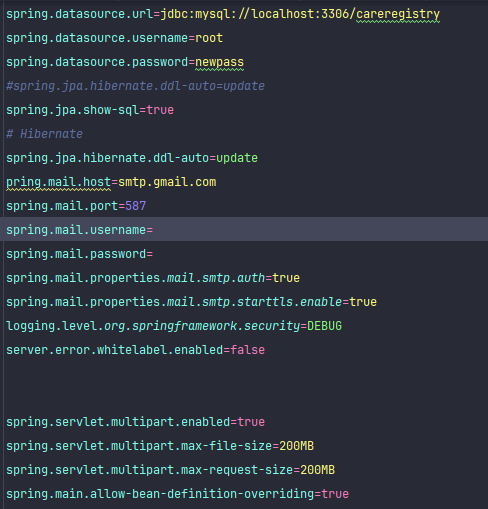
* Java
* Spring Boot (Version 3.2.0)
* Spring Dependency Management Plugin (Version 1.1.4)
* Thymeleaf
* Spring Web
* Spring Data JPA
* Spring Mail
* DKIM for JavaMail (de.vinado.spring:dkim-javamail:1.2.2)
* Spring Websocket
* Spring Security
* MySQL Connector Java (Version 8.0.23)
* Lombok
* Spring Boot DevTools
* JUnit Platform for testing

# 6. Getting Started

To run the Customer Care Registry Spring Boot project, follow these steps:

1. \*\*Database Configuration:\*\*

Update the database configuration in `application.properties` or `application.yml` file. Below is an example for MySQL:



## 2. Email Configuration:

Update the email configuration to enable email notifications. Example for Gmail:

Use this tutorial to set up your own SMTP service if you didn’t get the idea of this try searching SMTP for Spring Boot

<https://springjava.com/spring-boot/sending-mail-using-gmail-smtp-server-in-spring-boot>

## 3. Multipart File Configuration:

Configure maximum file size for file uploads:

spring.servlet.multipart.enabled=true

spring.servlet.multipart.max-file-size=200MB

spring.servlet.multipart.max-request-size=200MB

```

# 7. Installation

## 7.1 Prerequisites

Before you begin with the Customer Care Registry Spring Boot project, make sure you have the following prerequisites installed:

- Java Development Kit (JDK) 17 or later

- MySQL database server

- Your favorite Java Integrated Development Environment (IDE), such as IntelliJ or Eclipse

- Git for version control

- Gradle build tool

## 7.2 Installation

Follow these steps to set up and run the project locally:

1. Clone the Repository

git clone https://github.com/moxhadeel571/CustomerCare\_Registry\_SpringBoot.git

2. Open Project in IDE:

Open the project in your preferred IDE (IntelliJ or Eclipse).

3. Configure Database:

- Create a MySQL database.

- Update the database configuration in `src/main/resources/application.properties` with your database details.

4. Build and Run:

- Build the project using the following command:

./gradlew build

- Run the application:

./gradlew bootRun

5. Access the Application:

Once the application is running, access it in your web browser at [http://localhost:8080](http://localhost:8080).

6. Login:

Use the provided login features based on your assigned role (admin, care giver, customer, or team member).

7. Explore and Test:

Explore the various functionalities, such as care giver management, customer complaints, chat features, and team-related operations.

8. Contribute:

If you want to contribute or make changes, create a new branch, make your changes, and submit a pull request.

9. Shutdown:

Once done, gracefully shutdown the application.

Congratulations! You've successfully set up and run the Customer Care Registry Spring Boot project locally.

Note: The actual commands may vary based on your operating system. Adjust them accordingly.

Certainly! Here's a plain text version for the "7. Development" section of your project documentation:

# 7. Development

The development of the Customer Care Registry Spring Boot project involves several key components and controllers. Each section below provides an overview of the development aspects related to specific functionalities.

## 7.1 Care Giver Controller

The `CareGiverController` manages the interactions related to care givers. It includes endpoints for retrieving care giver information, displaying the care view, and saving form data. The controller is responsible for handling care giver-specific functionalities.

## 7.2 Chat Controller

The `ChatController` facilitates real-time communication between users. It includes endpoints for sending and receiving chat messages. The WebSocket functionality enables users to join the chat, send messages, and receive updates in real-time.

## 7.3 Recipient Controller

The `RecipientController` handles functionalities related to care recipients. This includes submitting complaints, checking complaint status, and downloading complaint-related files. The controller manages interactions with care recipients and their associated data.

## 7.4 Team Controller

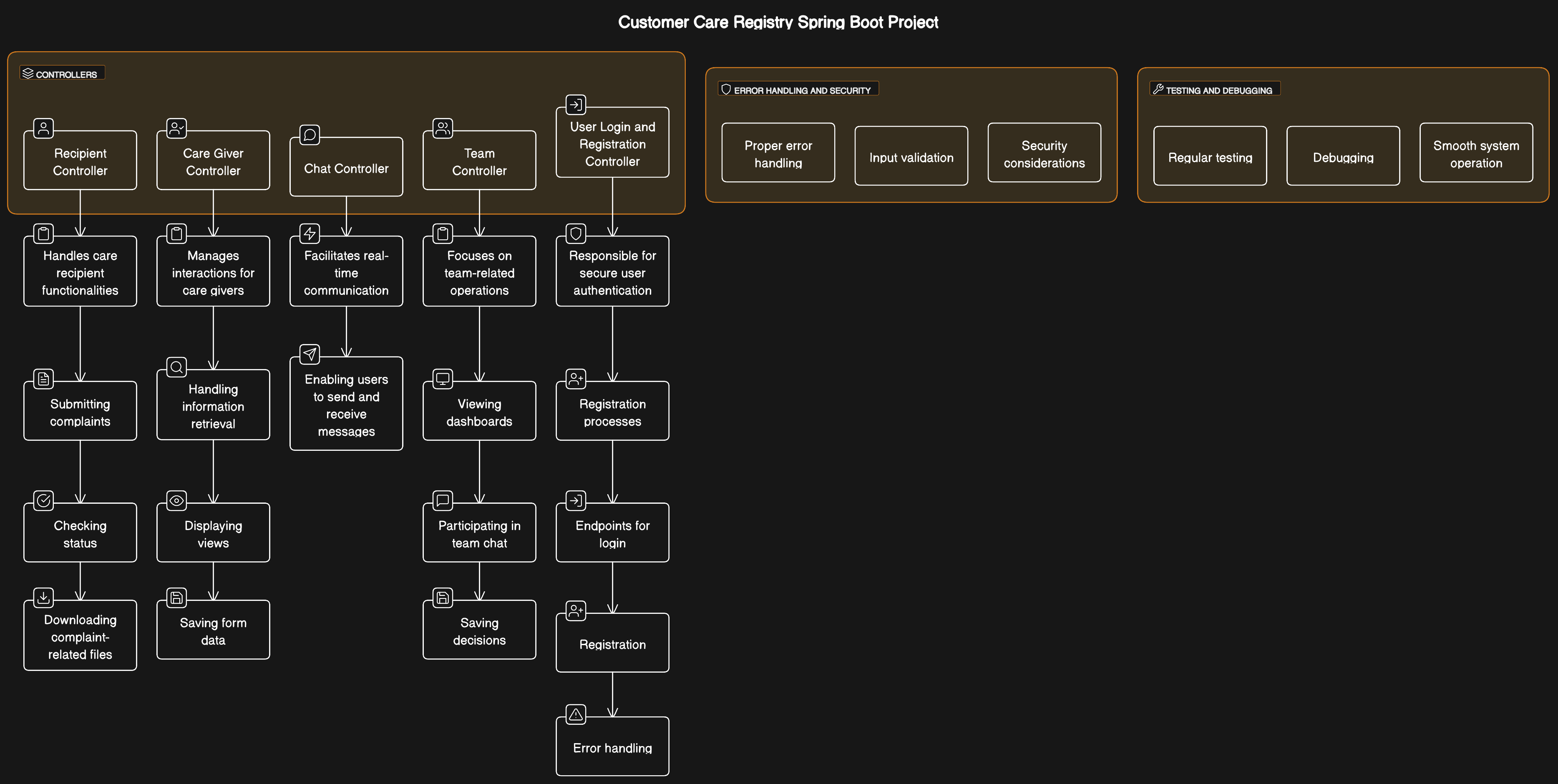
The `TeamController` focuses on functionalities related to the team. It includes endpoints for viewing the team dashboard, participating in team chat, and saving decisions related to care recipients. The controller facilitates team-specific operations.

## 7.5 User Login and Registration Controller

The `UserLRController` is responsible for user authentication and registration. It includes endpoints for user login, registration, and handling errors. The controller ensures secure user authentication and authorization.

These controllers work together to provide a comprehensive customer care registry system, covering various roles and functionalities. The development process involves implementing and testing these controllers to ensure the smooth operation of the entire system.

During development, attention should be given to proper error handling, input validation, and security considerations. Regular testing and debugging are essential to identify and address any issues that may arise during the development phase.



# 8. Models in Details

1. Care\_Giver:

- Entity representing care givers.

- Fields: `id`, `username`, `branchname`, `technicalname`, `address`, `city`, `state`, `country`, `phone\_Number`, `Experience`, `email`.

2. CareRecipient:

- Entity representing care recipients.

- Fields: `careRecipientId`, `name`, `productName`, `description`, `typeOfProduct`, `purchaseDate`, `images`, `contactNumber`, `emailAddress`, `address`, `incidentDateTime`, `complaintType`, `orderNumber`, `severityLevel`, `status`.

3. ChatMessage:

- Model representing a chat message.

- Fields: `type`, `content`, `sender`.

4. Email:

- Entity representing email messages.

- Fields: `id`, `from`, `to`, `subject`, `body`.

5. Image:

- Entity representing images associated with care recipients.

- Fields: `id`, `name`, `contentType`, `data`.

6. MessageType:

- Enum representing types of chat messages: `CHAT`, `JOIN`, `LEAVE`.

7. Team:

- Entity representing teams.

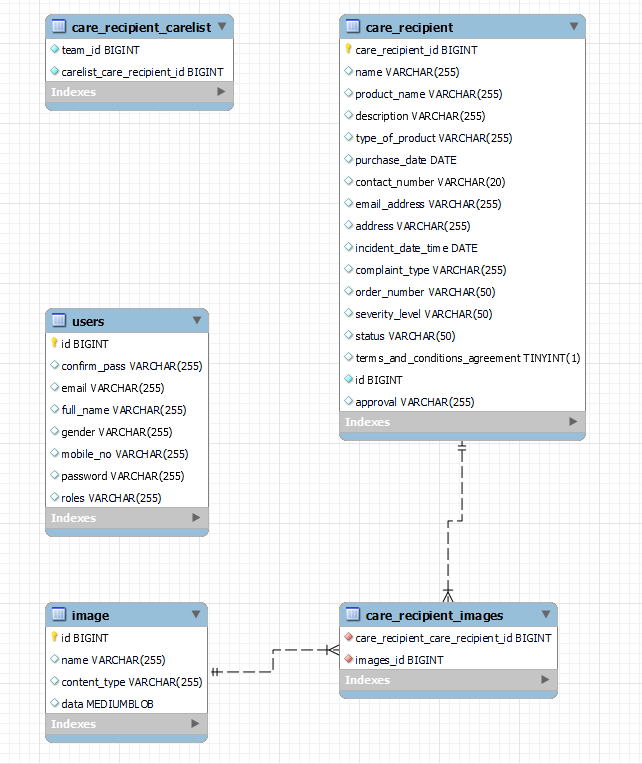
- Fields: `id`, `Approval`, `carelist`.

8. User:

- Entity representing users.

- Fields: `id`, `fullName`, `gender`, `confirmPass`, `mobileNo`, `password`, `email`, `roles`.

These models capture various aspects of your customer care registry system, including information about care givers, care recipients, chat messages, emails, images, teams, and users. Adjustments or additional details can be made based on specific project requirements.



# 

# 9. Repositories and Database Operations

1. **CareGiverRepository:**
   * Manages database operations for the **Care\_Giver** entities, allowing the retrieval, storage, and modification of caregiver information.
2. **ImageRepository:**
   * Handles database interactions for storing and retrieving **Image** entities. Provides methods for accessing image data, content type, and filenames.
3. **MailRepository:**
   * Manages database operations related to the storage and retrieval of **Email** entities. Includes methods for saving and querying email data.
4. **RecipientRepository:**
   * Facilitates database operations for **CareRecipient** entities. Allows the retrieval of file data, content type, and filenames associated with images. Also includes standard CRUD operations for care recipients.
5. **TeamRepository:**
   * Manages database operations for storing and retrieving **Team** entities. Provides methods for saving team decisions and accessing team-related data.
6. **UserRepository:**
   * Handles various database operations for **User** entities, including standard CRUD operations and a custom query to find a user by email.