Task-Handling-System Documentation

Sai Mallik Rameshwaram
Employee ID: 287969
saimallik.rameshwaram@ust.com

Jithin B Thomas
Employee ID: 288056 Jithin.BThomas@ust.com

November 15, 2024

Contents

Chapter 1

Introduction

1.1 Overview

The Task-Handling-System is designed to facilitate effective project and task management within teams. As organizations grow, managing multiple projects and the tasks associated with them can become complex and challenging. This system aims to streamline these processes, enabling better organization, collaboration, and productivity.

With features tailored to meet the needs of both managers and employees, the Task-Handling-System ensures that everyone can effectively track their responsibilities and project progress. By integrating role-based access control, the system maintains data security while allowing users to access relevant information according to their roles.

A key aspect of the Task-Handling-System is its focus on time-zone management. Given that teams may be distributed across different regions, the system stores all time-related data in UTC (Coordinated Universal Time) format. This approach simplifies scheduling and deadline management, ensuring that all team members have a consistent understanding of project timelines, regardless of their local time zones. Users can view and manage deadlines and tasks converted to their respective time zones, minimizing confusion and enhancing coordination.

The use of modern technologies, such as Angular for the front end and Spring Boot with PostgreSQL for the backend, guarantees a responsive and user-friendly experience. The system is built on a microservices architecture, promoting scalability and flexibility, which is crucial for adapting to changing project requirements.

1.2 Key Features

The Task-Handling-System includes the following primary features:

- **Project Management:** Easily create and manage different projects, each with its own description, timeline, and tasks. Keep track of project progress through important milestones and deadlines.

- **Task Assignment:** Managers can assign tasks to employees based on their roles and workloads. Each task can have due dates, priority levels, and detailed descriptions.
- **Role-Based Access Control:** Securely manage user roles, distinguishing between Employees and Managers. Employees can only see and manage their own tasks, while Managers can manage tasks for the whole team.
- **Task Status Tracking:** Track the status of tasks from start to finish, with options like To Do, In Progress, In Review, Completed, and Overdue. This helps everyone know how things are progressing.
- **Collaboration Tools:** Employees can comment on tasks, ask questions, and provide updates, which helps the team work together. Notifications keep everyone informed about new tasks and changes.
- **Performance Monitoring:** Managers can check how well employees are doing based on task completion and feedback. This provides insight into team productivity and areas that may need improvement.
- **Data Management:** Store all data in a structured way using PostgreSQL to ensure it's organized and easy to access. Use of standard options for priority and status keeps things consistent.
- **User-Friendly Interface:** The system has a simple and intuitive design built with Angular, making it easy for users to navigate. The design works well on various devices, so you can access it anywhere.
- **Secure User Authentication:** Implement security measures to manage user logins and access effectively. Use Spring Security to ensure that only authorized users can access certain features based on their roles.
- **RESTful API Integration:** Provide functionalities through RESTful APIs, allowing other applications to connect and use them easily. This makes it easier to build a flexible and scalable system.
- **Notification Service:** When an manager addes a task or when an employee completes a task both will receive the notification via mail and sms on update.

Conclusion: The Task-Handling-System offers an effective way to manage projects and tasks in different time-zones. while making sure that all users can easily navigate the system. It enhances teamwork and productivity by allowing clear visibility of tasks and responsibilities, ensuring everyone stays informed and engaged. .

Chapter 2

Use-Case Diagram

2.1 Diagram

The use case diagram for the Task-Handling-System illustrates the interactions between users (employees and managers) and the system's functionalities.

This diagram is essential as it clarifies user interactions and guides the development process by ensuring that all necessary functionalities are addressed. Additionally, it aids communication among stakeholders by providing a clear visual representation of how the system will operate. Overall, the use case diagram is a key tool in understanding the system's requirements and ensuring it meets user needs effectively.

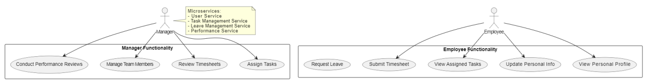

Figure 2.1: Use-Case Diagram

2.2 Key Use Cases

It highlights the primary tasks that users can perform, which include:

- **View Personal Profile**
- **Update Personal Info**
- **View Assigned Tasks**
- **Submit Timesheet**
- **Assign Tasks**
- **Review Timesheets**

- **Manage Team Members**
- **Notify Team Members and Manager on updates**

Chapter 3

System Architecture Overview

3.1 Microservice Architecture

The Task-Handling-System employs a microservice architecture, which divides the application into independent services that communicate over well-defined APIs. This design improves modularity, allowing each service to be developed, deployed, and scaled independently. Key services in this architecture include:

- **Eureka Registry:** A service discovery tool that allows microservices to find and communicate with each other.
- **Config Server:** Centralizes configuration management for all microservices, ensuring consistency and ease of updates.
- **API Gateway:** Acts as a single entry point for all client requests, routing them to the appropriate service while handling cross-cutting concerns such as authentication and logging.
- **User Service:** Manages user-related operations, including registration, authentication, and profile management.
- **Task Management Service:** Handles all operations related to tasks, such as creation, assignment, and status updates.
- **Security Service:** Implements security features, including role-based access control and token validation.
- **Notification Service:** Manages notifications to users regarding task assignments, updates, and other important events.

3.2 Service Interaction Flowchart

The interaction between micro services is crucial for the smooth operation of the Task-Handling-System. Below is a flowchart illustrating the typical service interaction in the system:

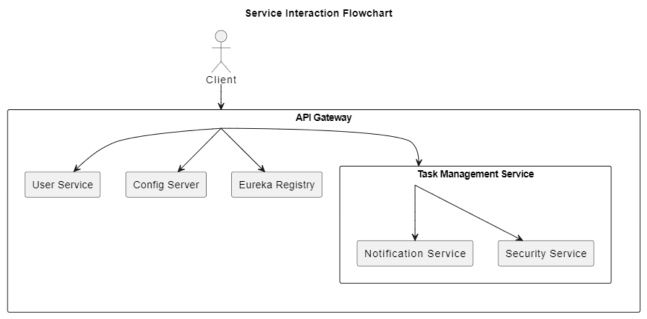

Figure 3.1: Service Interaction Flowchart

3.3 Class Descriptions

3.3.1 Project

Description: Represents a project within the task handling system. A project consists of various tasks that need to be completed and is managed by a specific manager.

Attributes:

- `projectId`: A unique identifier for the project (Long).
- `projectName`: The name of the project (String).
- `projectDescription`: A brief description of the project (String).
- `startDate`: The date when the project starts (LocalDate).
- `endDate`: The date when the project is scheduled to end (LocalDate).
- `managerId`: The ID of the manager responsible for the project (Long).
- `createdAt`: The timestamp indicating when the project was created (LocalDate).
- `updatedAt`: The timestamp indicating when the project was last updated (LocalDate).

Relationships:

- Has a **one-to-many** relationship with the `Task` class, meaning one project can contain multiple tasks.

3.3.2 Task

Description: Represents an individual task within a project. Tasks are assigned to employees and have specific attributes that define their state and requirements.

Attributes:

- `taskId`: A unique identifier for the task (Long).
- `taskTitle`: The title of the task (String).
- `taskDescription`: A detailed description of the task (String).
- `dueDateTime`: The due date and time for the task completion (LocalDateTime).
- `priority`: The priority level of the task (Priority enum).
- `status`: The current status of the task (Status enum).
- `employeeId`: The ID of the employee to whom the task is assigned (Long).
- `createdAt`: The timestamp indicating when the task was created (LocalDateTime).
- `updatedAt`: The timestamp indicating when the task was last updated (LocalDateTime).
- `completedAt`: The timestamp indicating when the task was completed (LocalDateTime).

Relationships:

- Has a **many-to-one** relationship with the `Project` class, meaning each task is associated with a single project.

3.3.3 Priority (Enum)

Description: Enumerates the different priority levels that can be assigned to tasks, helping to classify their urgency.

Values:

- `LOW`: Indicates that the task is of low importance.
- `MEDIUM`: Indicates that the task is of moderate importance.
- `HIGH`: Indicates that the task is of high importance.

3.3.4 Status (Enum)

Description: Enumerates the various statuses a task can have throughout its lifecycle, reflecting its current state.

Values:

- `TODO`: Indicates that the task is yet to be started.
- `IN_PROGRESS`: Indicates that work on the task is currently underway.
- `IN_REVIEW`: Indicates that the task is completed and is under review.
- `COMPLETED`: Indicates that the task has been finished.
- `OVERDUE`: Indicates that the task's due date has passed without completion.

3.3.5 Notification

Description: Represents a notification sent to employees regarding task assignments, updates, or other system events. Notifications ensure that users are kept informed about their tasks and responsibilities.

Attributes:

- `notificationId`: A unique identifier for the notification (Long).
- `message`: The content of the notification (String).
- `timestamp`: The date and time when the notification was created (LocalDateTime).
- `employeeId`: The ID of the employee who is the recipient of the notification (Long).

Relationships:

- Sends notifications to employees related to task assignments, status updates, and other relevant system events.

Chapter 4

ER Diagram

4.1 Overview

The ER diagram for the **Task Handling System** illustrates the entities, their attributes, and the relationships between them.

4.1.1 Key Entities

- **Project**
  - **Attributes**:
    - `projectId` (PK), `projectName`, `projectDescription`, `startDate`, `endDate`, `managerId`, `createdAt`, `updatedAt`
  - **Relationship**: One project can have many tasks (One-to-Many with Task).
- **Task**
  - **Attributes**:
    - `taskId` (PK), `taskTitle`, `taskDescription`, `dueDateTime`, `priority`, `status`, `employeeId`, `createdAt`, `updatedAt`, `completedAt`
  - **Relationship**: Each task belongs to one project (Many-to-One with Project) and can generate notifications (One-to-Many with Notification).
- **Priority (Enum)**
  - Values: LOW, MEDIUM, HIGH
- **Status (Enum)**
  - Values: TODO, IN_PROGRESS, IN_REVIEW, COMPLETED, OVERDUE

- **Notification**
  - **Attributes**:
    - `notificationId` (PK), `message`, `timestamp`, `employeeId`
  - **Relationship**: Notifications are associated with tasks and sent to employees (Many-to-One with Employee).

4.1.2 Relationships Summary

- **Project to Task**: One-to-Many
- **Task to Employee**: Many-to-One
- **Task to Notification**: One-to-Many

This concise overview captures the essential components of the ER diagram, highlighting how different entities interact within the system.

4.2 Diagram

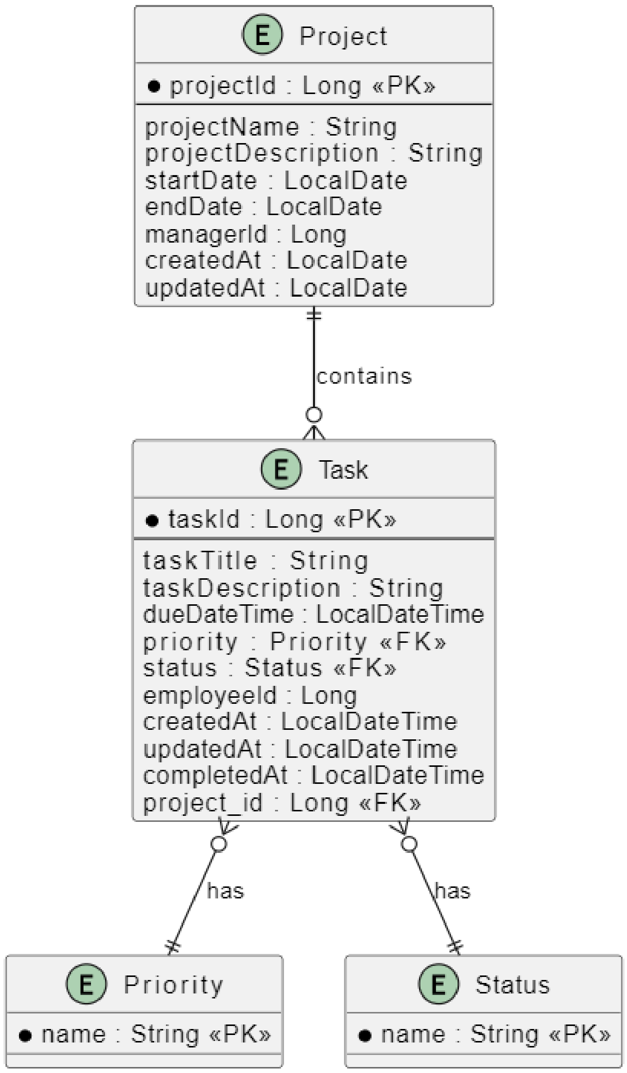

Figure 4.1: ER Diagram

Chapter 5

Sequence Diagram

5.1 Overview

The sequence diagram for the **Task Handling System** illustrates the interaction between various components during a typical task assignment process.

5.1.1 Key Participants

- **Client**
  - Represents the user interface through which managers and employees interact with the system.
- **API Gateway**
  - Acts as a single entry point for requests, directing them to appropriate microservices.
- **Eureka Registry**
  - Manages service discovery, allowing the API Gateway to locate available services.
- **Config Server**
  - Provides configuration settings to the microservices at runtime.
- **User Service**
  - Manages user information and authentication processes.
- **Task Management Service**
  - Handles task creation, updates, and retrieval for both employees and managers.

- **Notification Service**
  - Sends notifications to users based on task-related events.

5.1.2 Sequence of Events

1. The **Client** initiates a request to assign a task.
2. The request is sent to the **API Gateway**.
3. The **API Gateway** queries the **Eureka Registry** to find the **User Service** and **Task Management Service**.
4. The **API Gateway** calls the **User Service** to validate the manager's credentials.
5. Upon successful validation, the **API Gateway** sends the task assignment request to the **Task Management Service**.
6. The **Task Management Service** creates the task and updates its records.
7. A notification is generated and sent to the **Notification Service**.
8. The **Notification Service** sends a notification to the relevant employee regarding the assigned task.

This sequence diagram captures the interactions and flow of control between various components in the Task Handling System during the task assignment process, ensuring that all necessary steps are covered for effective communication and operation.

5.2 Diagram

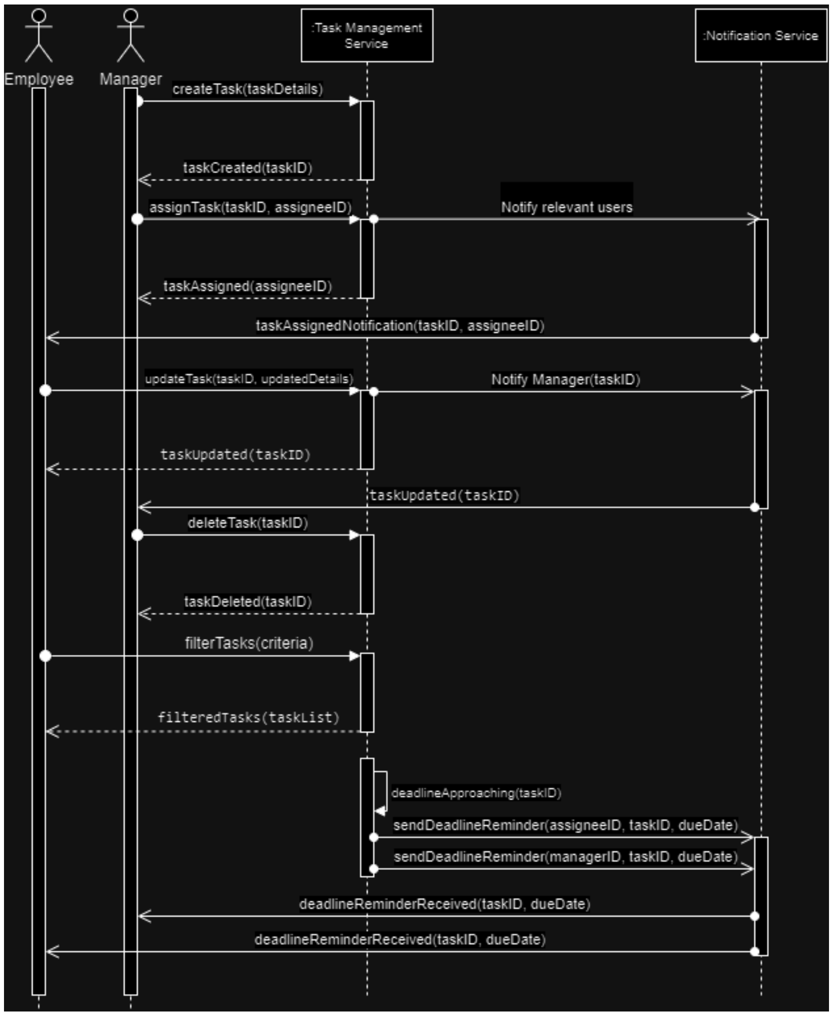
Employee
Manager
:Task Management Service
:Notification Service
createTask(taskDetails)
taskCreated(taskID)
assignTask(taskID, assigneeID)
Notify relevant users
taskAssigned(assigneeID)
taskAssignedNotification(taskID, assigneeID)
updateTask(taskID, updatedDetails)
Notify Manager(taskID)
taskUpdated(taskID)
taskUpdated(taskID)
deleteTask(taskID)
taskDeleted(taskID)
filterTasks(criteria)
filteredTasks(taskList)
deadlineApproaching(taskID)
sendDeadlineReminder(assigneeID, taskID, dueDate)
sendDeadlineReminder(managerID, taskID, dueDate)
deadlineReminderReceived(taskID, dueDate)
deadlineReminderReceived(taskID, dueDate)

Figure 5.1: Sequence Diagram for Task Assignment Workflow

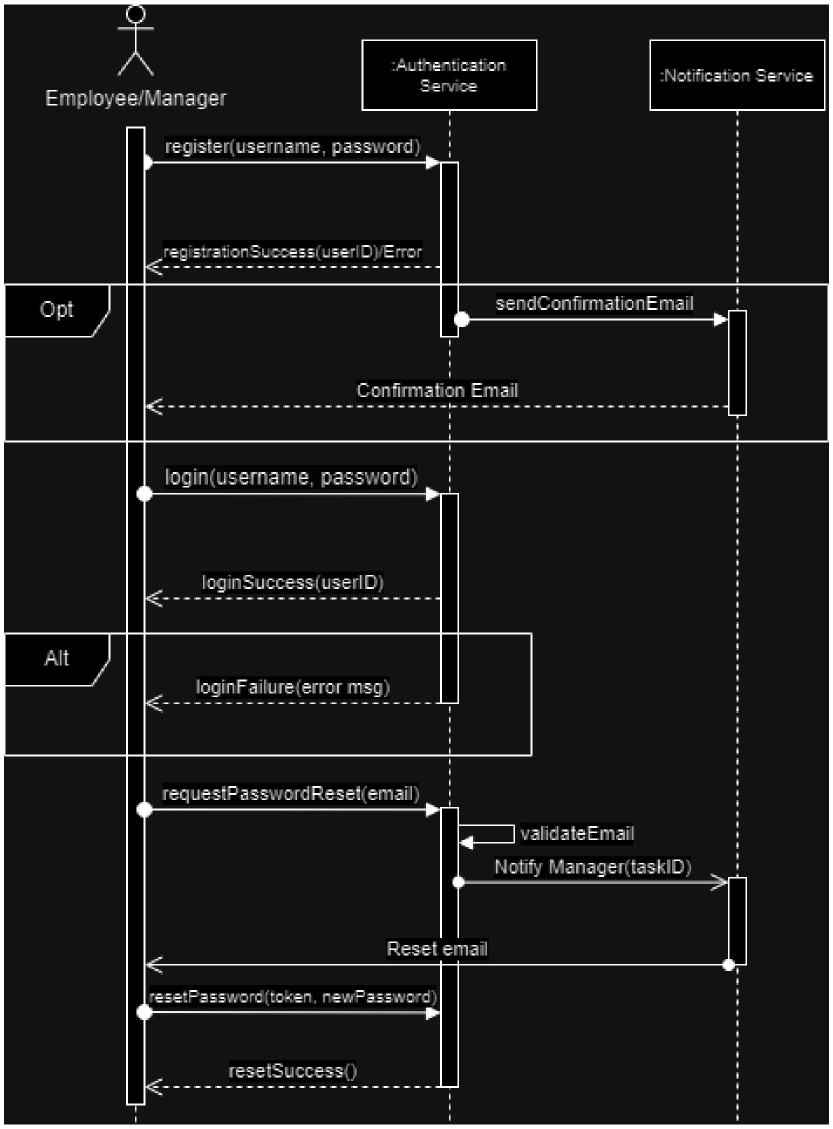

Figure 5.2: Sequence Diagram for User-Authentication

Chapter 6

Package Diagram

6.0.1 Overview

The package diagram for the **Task Handling System** provides a high-level view of the system's architecture, showcasing how different packages are organized and their dependencies.

6.0.2 Key Packages

- **User Management Package**
  - Responsible for handling user-related functionalities, including registration, authentication, and role management.
- **Project Management Package**
  - Contains classes and services for managing projects, including project creation, updates, and status tracking.
- **Task Management Package**
  - Manages tasks associated with projects, including task creation, assignment, updates, and retrieval.
- **Notification Package**
  - Handles the generation and sending of notifications to users regarding task assignments, updates, and other relevant events.
- **Security Package**
  - Implements security features, including authentication, authorization, and role-based access control for different users.
- **Configuration Package**

    - Manages configuration settings and external properties required by other packages to function correctly.

- **Eureka Registry Package**
    - Facilitates service discovery, allowing different services to register themselves and discover other services in the system.

6.0.3 Dependencies

- The **API Gateway** package interacts with all other packages to route requests and manage service interactions.
- The **Notification** package depends on the **Task Management** package to send updates based on task events.
- The **User Management** package is essential for providing user information to the **Security** package for authentication and authorization processes.

This package diagram provides a structured view of how various components are organized within the Task Handling System, helping to understand the system's modular architecture and the interactions among different packages.

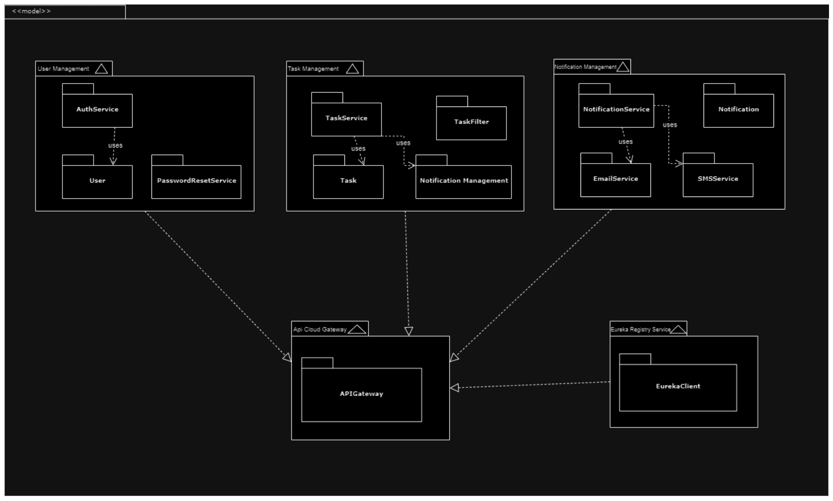

Figure 6.1: Sequence Diagram for User-Authentication

Chapter 7

Class Diagrams

The class diagram for the **Task Handling System** illustrates the structure of the system by showing the system's classes, their attributes, methods, and the relationships among the classes.

7.0.1 Key Classes

- **Project**
  - **Attributes:**
    * projectId: Long
    * projectName: String
    * projectDescription: String
    * startDate: LocalDate
    * endDate: LocalDate
    * managerId: Long
    * createdAt: LocalDate
    * updatedAt: LocalDate
    * tasks: List¡Task¿
  - **Methods:**
    * getters and setters for each attribute
- **Task**
  - **Attributes:**
    * taskId: Long
    * taskTitle: String
    * taskDescription: String
    * dueDateTime: LocalDateTime
    * priority: Priority

    * status: Status
    * employeeId: Long
    * createdAt: LocalDateTime
    * updatedAt: LocalDateTime
    * completedAt: LocalDateTime
    * project: Project
  - **Methods:**
    * getters and setters for each attribute

- **Priority (Enum)**
  - **Values:** LOW, MEDIUM, HIGH
- **Status (Enum)**

- **Values:** TODO, IN$_{P}ROGRESS, IN_{R}EVIEW, COMPLETED, OVERDUE$

- **User**
  - **Attributes:**
    * userId: Long
    * username: String
    * password: String
    * role: String (e.g., Manager, Employee)
  - **Methods:**
    * getters and setters for each attribute
- **Notification**
  - **Attributes:**
    * notificationId: Long
    * message: String
    * recipientId: Long
    * timestamp: LocalDateTime
  - **Methods:**
    * getters and setters for each attribute

7.0.2 Relationships

- The **Project** class has a one-to-many relationship with the **Task** class, as a project can have multiple tasks.
- The **User** class can be associated with tasks via the **employeeId** attribute in the **Task** class, indicating which user the task is assigned to.
- The **Notification** class is related to the **User** class, as notifications are sent to users based on task-related events.

This class diagram provides a detailed view of the entities in the Task Handling System, their attributes, methods, and how they interact with each other.

Chapter 8

API Documentation

8.1 User Service Endpoints

List the API endpoints provided by the User Service, including request and response examples.

8.2 Task Management Service Endpoints

List the API endpoints for task management.

8.3 Role Management Service Endpoints

Describe the endpoints for managing user roles and permissions.

Chapter 9

Role Management

9.1 Overview

Provide an overview of the role-based access control system in Task-Handling-System.

9.2 Roles and Permissions

Detail the roles available (Manager and Employee) and their respective permissions.

9.3 Security Implementation

Explain how security is implemented, including annotations for role-based access.

9.4 Role Assignment

Describe the role assignment process during user registration.

Chapter 10

Deployment and Configuration

10.1 Deployment Steps

Provide instructions for deploying the Task-Handling-System.

10.2 Service Configurations

Detail any configuration settings required for each microservice.

Appendix A

Appendices

A.1 Glossary

Define any specific terms used in the documentation.

A.2 References

List any references used.

A.3 Diagrams Index

Include an index of all diagrams used in the document for quick reference.