Department of Computer Science
Technical University of Cluj-Napoca

Stack Overflow Assignment

Name:Stroie Anca Gabriela
Group:30432
Email:Stroie.Io.Anca@student.utcluj.ro

Contents

Chapter 1

Introduction

1.1 Overview

StackOverflow is a popular online platform where users can ask questions, provide answers, and engage in discussions on various topics related to software development, programming, and technology. Our project aims to implement a simplified version of StackOverflow, focusing on core features such as asking questions, providing answers, voting on content, and user moderation.

1.2 Features

- **Question Management**: Users can ask questions, providing details such as title, text, and tags. Each question is associated with an author, creation date & time, and may include a picture. Users can edit or delete their own questions. Questions are displayed sorted by creation date, with the most recent ones appearing first. Filtering options are provided based on tags, text search, users, or the user's own questions.

- **Answering Questions**: Questions may be answered by any user, including the original author. Each answer includes details such as author, text, creation date & time, and may include a picture. Users can edit or delete their own answers. When viewing a question, the list of answers is displayed along with the question details.

- **Voting System**: Users can vote on questions and answers (upvote and downvote, like and dislike). Each user can only vote once on each question or answer. Vote count for each question or answer is displayed, calculated based on upvotes and downvotes/likes and dislikes. Answers for a question are sorted by their vote count, with the highest-voted answers displayed first.

- **User Score**: Users earn points based on upvotes received for their questions and answers. Points are deducted for downvotes received on their questions or answers, as well as for downvoting others' answers. The user score is displayed next to the author's name on questions and answers, reflecting their overall contribution and reputation.

- **Moderator Privileges**: Moderators have special privileges to manage content and users on the platform. They can remove inappropriate questions or answers, edit any content, and ban users from the site in case of misconduct. Banned users receive notifications via email and SMS and are restricted from accessing the platform.

Chapter 2

Tech Stack

Ourapplication is built using a combination of technologies that ensure reliability, scalability, and ease of development. The tech stack comprises the following components:

- **Java Spring Framework**: Our backend is developed using the Java Spring Framework, a robust and widely adopted framework for building enterprise-grade applications. Spring provides extensive support for building RESTful APIs, handling database operations, and implementing business logic.

- **PostgreSQL**: We use PostgreSQL, an open-source relational database management system, for storing and managing the data of our hotel management system. PostgreSQL offers excellent reliability, scalability, and performance, making it an ideal choice for our application.

- **Postman**: For API development, testing, and documentation, we rely on Postman, a popular collaboration platform for API development. Postman provides a user-friendly interface for sending requests, testing endpoints, and generating API documentation, streamlining our API development workflow.

Our carefully selected tech stack enables us to develop a scalable, reliable, and efficient hotel management system that meets the needs of our users and stakeholders. By leveraging these technologies, we aim to deliver a high-quality software solution that enhances hotel management processes and improves guest experiences.

Chapter 3

SW Architecture

The software architecture of our hotel management system is structured around four key layers: Controller, Entity, Repository, and Service. Each layer plays a distinct role in the system, contributing to its functionality, maintainability, and scalability.

Controller Layer:

- Manages incoming HTTP requests and routes them to appropriate service methods.

- Validates user input and ensures proper handling of request parameters.

- Orchestrates the interaction between the frontend user interface and the backend logic.

Entity Layer:

- Defines the structure and behavior of domain entities such as questions, users, and answers .

- Represents real-world concepts as Java objects annotated with JPA annotations.

- Ensures data integrity and consistency by enforcing constraints and relationships.

Repository Layer:

- Encapsulates data access logic and CRUD operations for interacting with the database.

- Provides a clean and consistent interface for querying and manipulating entity data.

- Implements database interactions using Spring Data JPA interfaces and annotations.

Service Layer:

- Contains the core business logic and orchestrates interactions between multiple entities.

- Coordinates complex business operations such as room booking, inventory management, and payment processing.

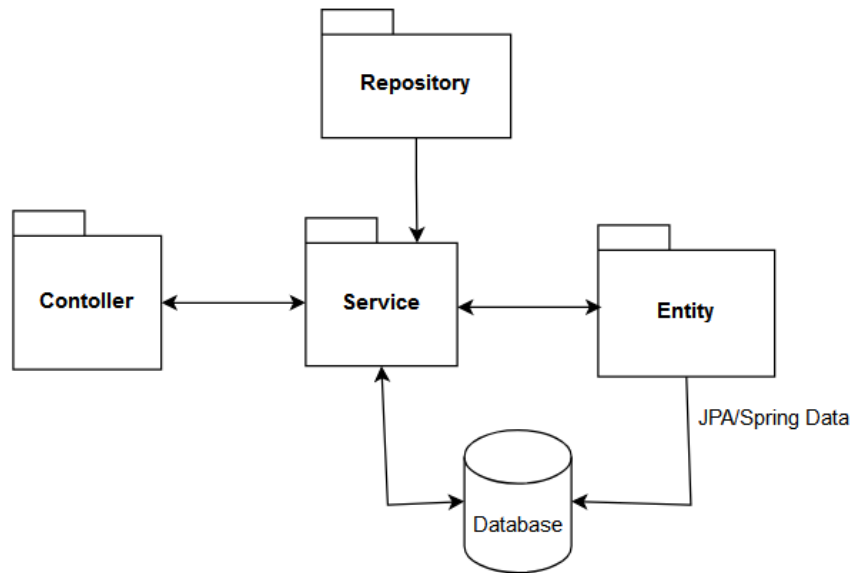- Enforces business rules and policies to ensure accurate and consistent behavior of the system.

Figure 3.1: Package Diagram.

Chapter 4

Functional Requirements

- **User Authentication and Authorization:**
  - Users can register and log in securely.
  - Only logged-in users can access system features.

- **Question Management:**
  - Users can ask, edit, and delete questions.
  - Questions display author, creation date, and can be filtered by tags or search.

- **Answering Questions:**
  - Users can answer, edit, and delete answers.
  - Answers display author, creation date, and are listed under questions.

- **Voting System:**
  - Users can upvote or downvote questions and answers.
  - Vote count is displayed and answers are sorted by vote count.

- **User Score Calculation:**
  - Users earn points for upvotes and lose points for downvotes.

- **Moderator Privileges:**
  - Moderators can manage content and users, including banning users.
  - Banned users receive notifications and are restricted from accessing the platform.

Chapter 5

Non-Functional Requirements

- **Performance:** The system must handle a large number of concurrent users without significant degradation in response time.

- **Security:** User data, including passwords, must be securely encrypted and stored. Access to sensitive features must be restricted based on user roles.

- **Reliability:** The system must be available and operational 24/7 with minimal downtime for maintenance or upgrades.

- **Scalability:** The system must be able to scale horizontally to accommodate increased user load by adding more servers or resources.

- **Usability:** The user interface must be intuitive and easy to navigate, with clear instructions and error messages.

- **Compatibility:** The system should be compatible with a wide range of web browsers and devices to ensure accessibility for all users.

- **Maintainability:** The codebase must be well-organized and documented, allowing for easy maintenance, updates, and bug fixes by developers.

Chapter 6

Diagrams

6.1 Database

The database for our application is designed to store and manage data related to various aspects, including questions, answers, users. The database schema consists of several tables, each representing a specific entity or relationship within the system.

Figure 6.1: Database Diagram
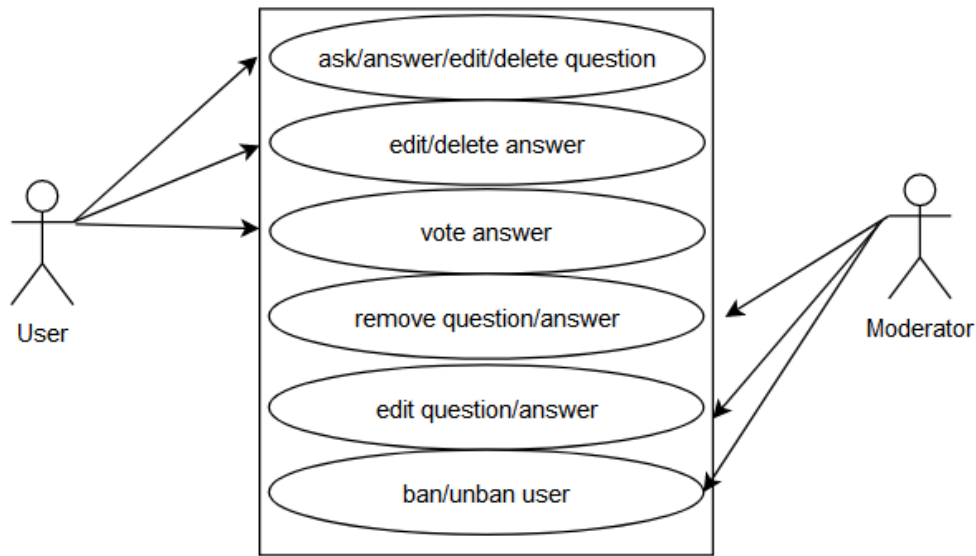
6.2   Use Case Diagram

Figure 6.2: Use Case Diagram.

User:

- **Ask Question:** Allows the user to ask a question by providing title, description, and tags.

- **Answer Question:** Enables the user to answer a question with text response.

- **Edit Question:** Allows the user to edit their own questions.

- **Delete Question:** Enables the user to delete their own questions.

- **Edit Answer:** Allows the user to edit their own answers.

- **Delete Answer:** Enables the user to delete their own answers.

- **Vote Question/Answer:** Allows the user to vote on questions and answers (upvote/-downvote).

Moderator:

- **Remove Question/Answer:** Allows the moderator to remove inappropriate questions or answers.

- **Edit Question/Answer:** Enables the moderator to edit any question or answer on the site.

- **Ban User:** Allows the moderator to ban users from the site indefinitely in case of bad behavior.

- **Unban User:** Enables the moderator to unban previously banned users.
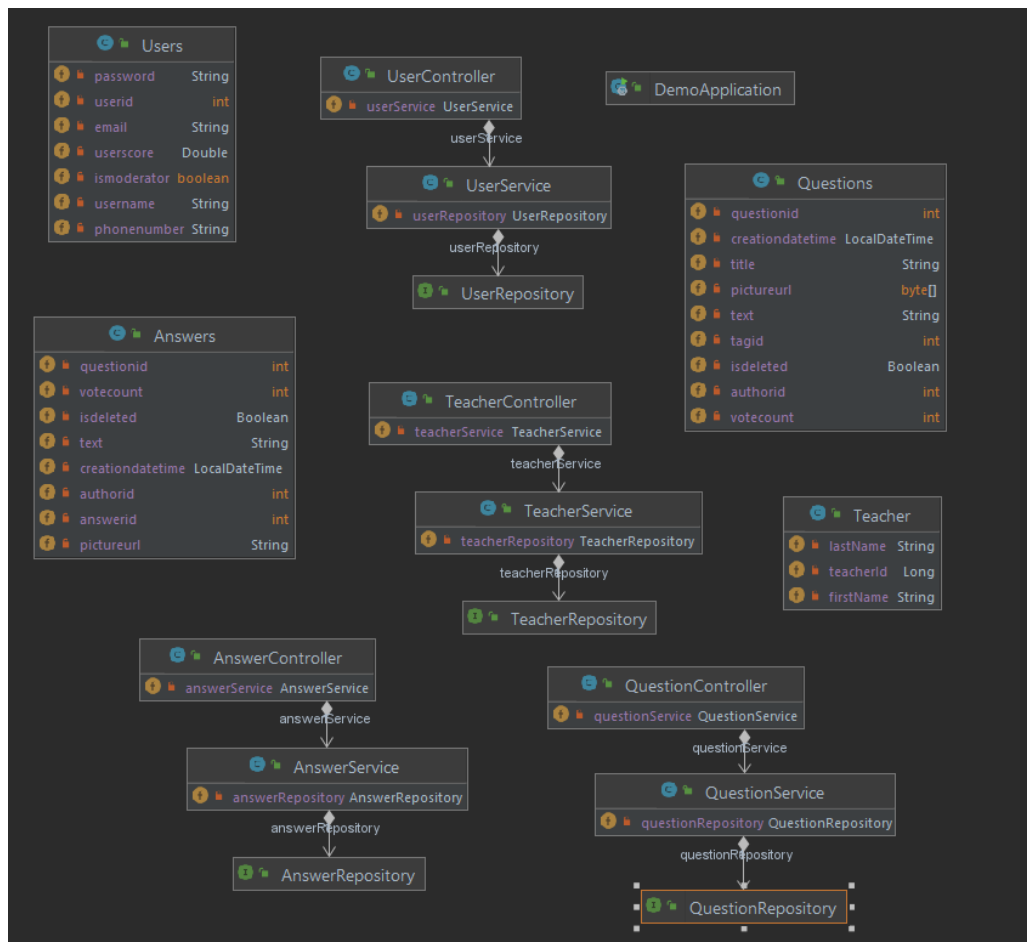
6.3 Class Diagram

Figure 6.3: Class Diagram.

Bibliography