USOF project

Q&A Web app

Developed by: first-year student, group KN-724c
Kostiantyn LAZEBNYI
NATIONAL TECHNICAL UNIVERSITY "KHARKIV POLYTECHNIC INSTITUTE"
Kharkiv 2025

Agenda

Introduction

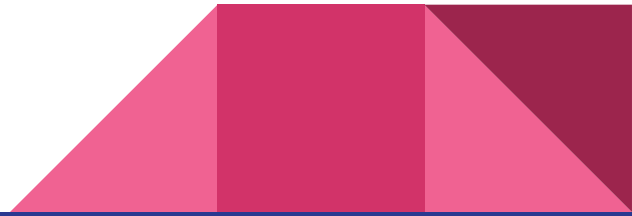
Functional Requirements

Non-Functional Requirements

DB diagram

UML Diagrams (Use-Case, Activity, Component)

Conclusion / Future Work



Introduction

The USOF *Answer & Question System* is a web-based platform that allows users to ask questions, provide answers, and interact within a community.
The system supports both frontend and backend parts, handling user authentication, content management, and administrative moderation.
Users can browse content without registration, while authorized users can create and manage their own questions and answers.
An administrator has extended privileges to maintain platform integrity and user compliance.
A third-party mail server is integrated for two-factor registration and password recovery via verification codes.

Functional Requirements

1. **User Registration and Login:**
   - Users must be able to **register** with a unique username and email.
   - Users must be able to **log in** using username/email and password.
   - The system must support **two-factor authentication** during registration.
2. **Question Management:**
   - Authorized users can **create, edit, and delete** their own questions.
   - Questions must include **title, description, and category(es)**.
   - All users can **view questions** with associated answers and brief user info.
3. **Answer Management:**
   - Authorized users can **create, edit, and delete** their own answers.
   - All users can **view answers** with brief author information.
   - Answers must be linked to a specific question.

Functional Requirements

4. **Likes and Dislikes:**

   - Users can **like or dislike** questions and answers.
   - Each user can only vote **once per item**.
   - The system must update the **vote count in real-time**.

5. **User Profile Management:**

   - Authorized users can **view and partially edit** their profile information.
   - Users can **change password** and manage account settings.
   - Admin users can **manage other users** (view/edit/delete limited info).

Functional Requirements

6. Email Verification and Notifications:

- Users must receive **email verification codes** during registration and password recovery.
- The system must validate codes and **activate accounts** only after successful verification.

7. Search and Filtering:

- Users must be able to **search questions** by keywords, categories or authors.
- The system should allow **filtering answers** by date, popularity, or relevance.

Functional Requirements

8. **Admin Functions:**
    - Admins can **moderate content**, including questions, answers, and user profiles.
    - Admins can **delete inappropriate content** or block users if necessary.
9. **System Logging and Auditing:**
    - All important actions (registration, login, question creation, votes) must be **logged**.
    - Admins can **view logs** for audit purposes.

Non-Functional Requirements

Performance:

- **Response Time:**
  The system must respond to user requests within **2 seconds** under a load of up to **1000 concurrent users**.
- **Page Load Time:**
  Pages should load within **3 seconds** on a **10 Mbps** internet connection.
- **Throughput:**
  The system should handle at least **500 requests per second** during peak usage.

Non-Functional Requirements

Cross-Browser Compatibility:

- Fully compatible with **4 most popular browsers**: Chrome, Edge, Firefox, Safari.

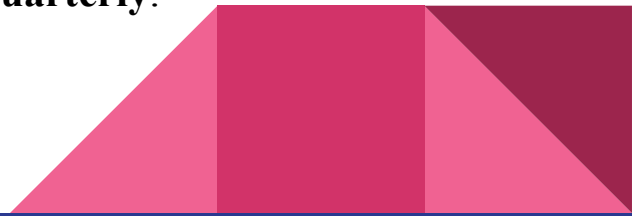
UI Design / Color Scheme:

- **Backgrounds:** Dark gray
- **Font color:** White
- **Highlight colors:** Red, green, or blue elements must not exceed **10% of total UI**

Non-Functional Requirements

Security:

- **Password Storage:**
  User passwords must be stored using **AES-256 encryption**.
- **Authentication:**
  Implement **two-factor authentication** for all user accounts.
- **Data Transmission:**
  All data transmitted between the client and server must be encrypted using **TLS 1.2 or higher**.
- **Vulnerability Testing:**
  Conduct regular security audits and penetration testing at least **quarterly**.



Non-Functional Requirements

Reliability:

- **Uptime:**
  The application must maintain **99.9% uptime** annually, equating to no more than **8.76 hours of downtime per year**.
- **Error Recovery:**
  In the event of a failure, the system should recover automatically within **5 minutes** without data loss.
- **Backup Frequency:**
  Perform data backups every **24 hours**, with the ability to restore from backups within **1 hour**.

Non-Functional Requirements

Usability:

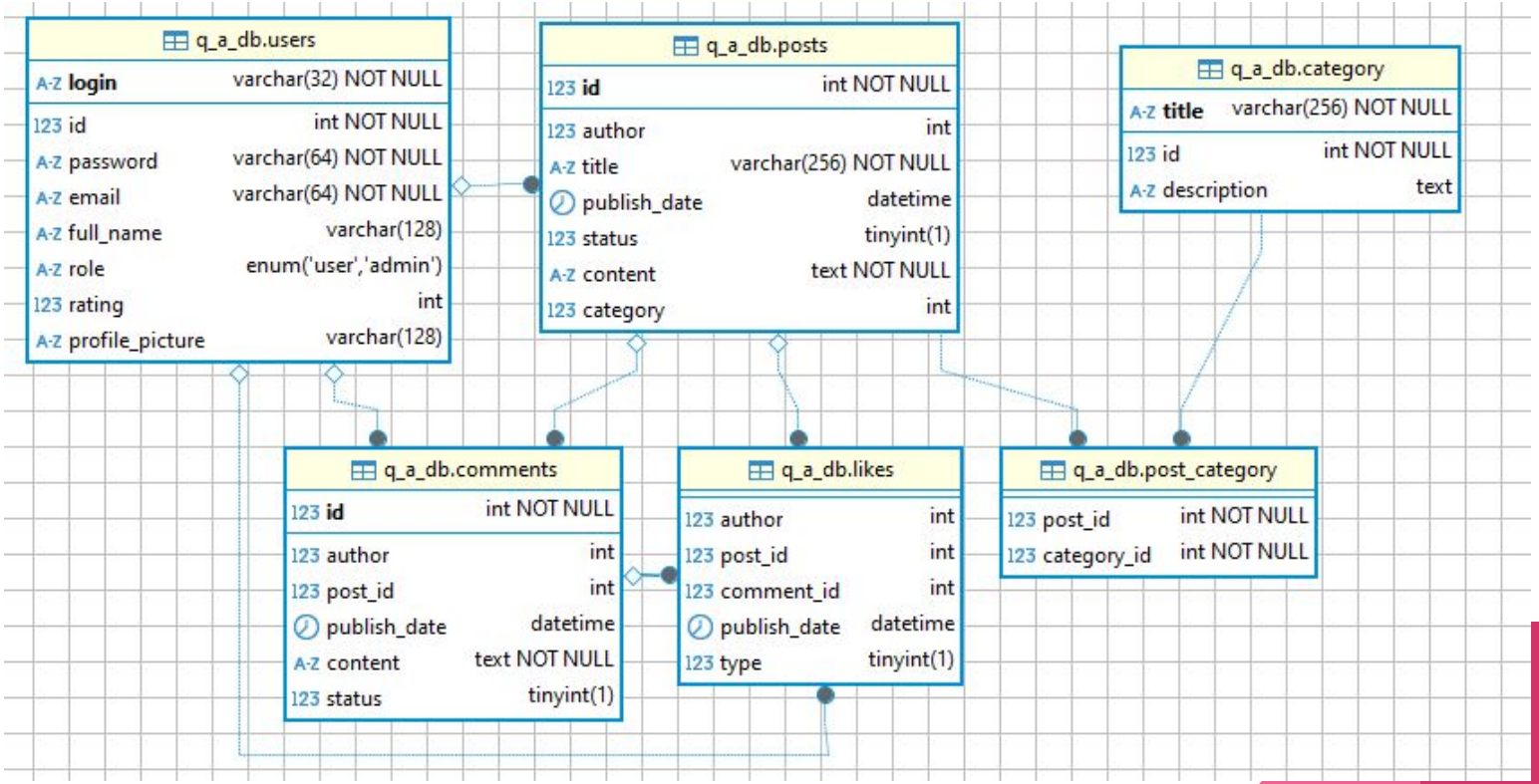
- **User Interface:**
  The user interface should be intuitive and require no more than **2 hours** of training for new users.
- **Accessibility:**
  Ensure compliance with **WCAG 2.1 Level AA** accessibility standards.
- **User Feedback:**
  Conduct usability testing with a minimum of **10 users** and implement changes based on feedback within **2 weeks**.

Non-Functional Requirements

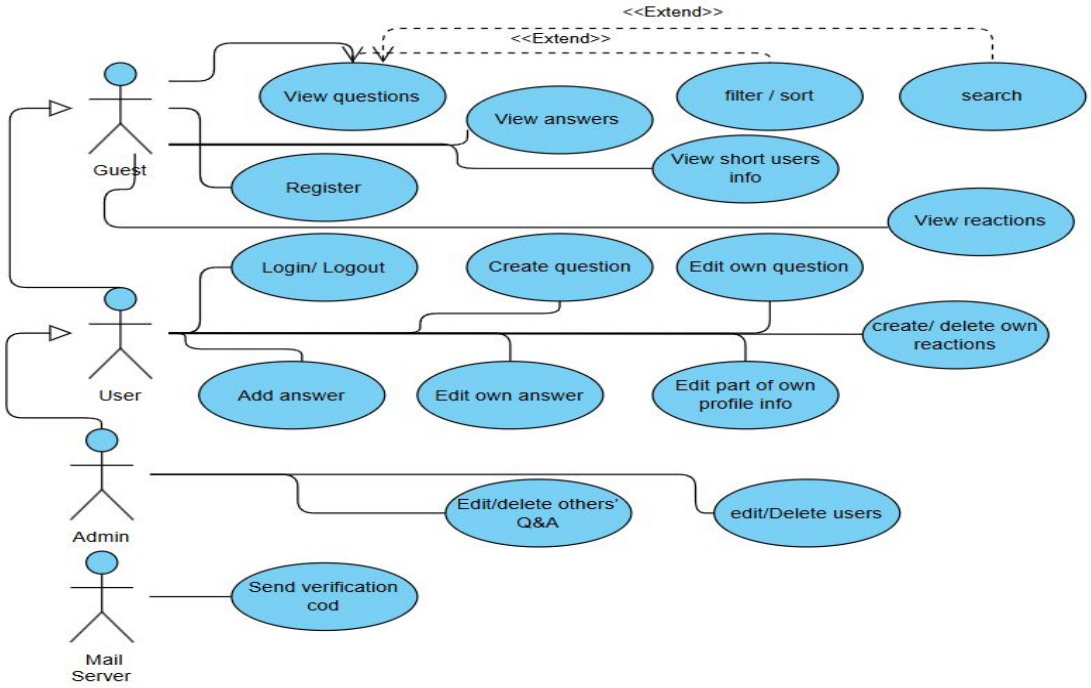
Maintainability:

- **Code Quality:**
  Maintain a code quality score of at least **80%** as measured by static code analysis tools.
- **Documentation:**
  Provide comprehensive documentation for all APIs, with updates made within **1 week** of any changes.
- **Bug Resolution:**
  Resolve critical bugs within **24 hours** and non-critical bugs within **5 business days**.

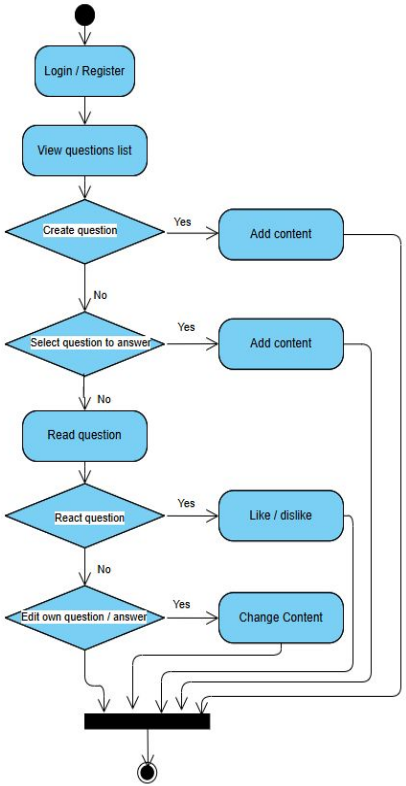
DB diagram


q_a_db.users
login varchar(32) NOT NULL
id int NOT NULL
password varchar(64) NOT NULL
email varchar(64) NOT NULL
full_name varchar(128)
role enum('user','admin')
rating int
profile_picture varchar(128)
q_a_db.posts
id int NOT NULL
author int
title varchar(256) NOT NULL
publish_date datetime
status tinyint(1)
content text NOT NULL
category int
q_a_db.category
title varchar(256) NOT NULL
id int NOT NULL
description text
q_a_db.comments
id int NOT NULL
author int
post_id int
publish_date datetime
content text NOT NULL
status tinyint(1)
q_a_db.likes
author int
post_id int
comment_id int
publish_date datetime
type tinyint(1)
q_a_db.post_category
post_id int NOT NULL
category_id int NOT NULL

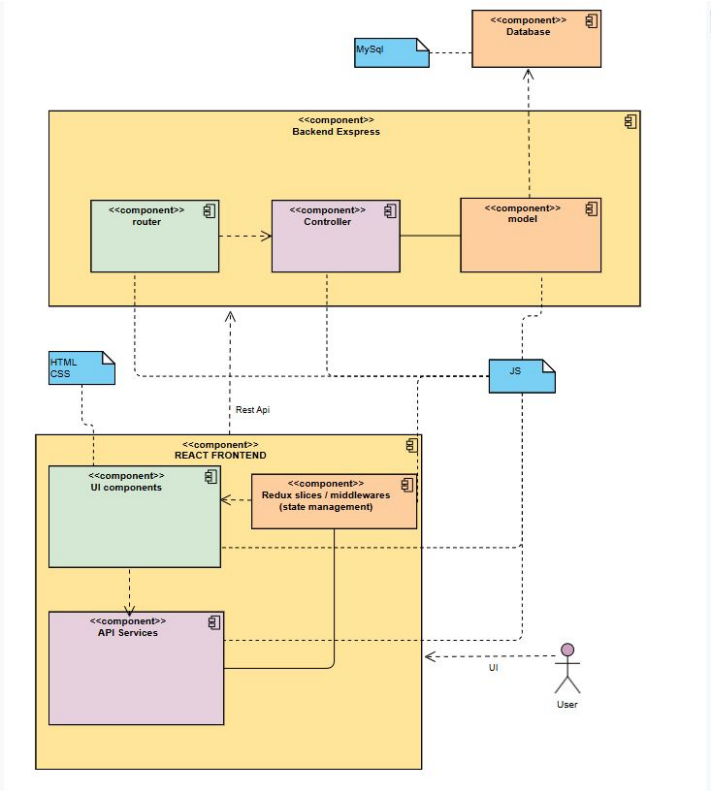
Use-Case diagram



Activity diagram of main functionality



Component diagram Highest level



Conclusion / Future Work

Although applications of this type may seem somewhat traditional, they can serve as an excellent module within larger platforms, such as our Innovation Campus learning system. They can also function as a standalone component to reduce the load on the main system.

The next step is to develop the administrative part of the application. Future improvements may include expanding user roles, such as adding a moderator, and incorporating user feedback to better align the system with user needs.