**Heaven’s Light Is Our Guide**

**Rajshahi University of Engineering & Technology**

**Department of Computer Science & Engineering**

A logo with text and symbols

Description automatically generated with medium confidence

**Course Code:** CSE 3100

**Course Title:** Software Engineering Sessional

**Title:**

|  |  |
| --- | --- |
| **Submitted By**  Name: Rushnan Reaz  Roll: 2003101  Section: B, Session: 2020-2021 | **Submitted To**  Md. Mohiuddin Ahmed  Assistant Professor  Department of CSE, RUET |

**Submission Date:** 31.12.2024

## 1. Project Overview

**Exchange** is a web application designed to facilitate question and answer interactions among users. It includes features such as user authentication, question posting, answering, commenting, and voting on questions and answers. The application is built using React.js for the frontend and Node.js/Express for the backend.

## 2. Architecture

### 2.1 Frontend (React.js)

* **Component Structure**: The application is divided into multiple reusable components.
* **State Management**: Managed using React’s useState and useContext hooks.
* **Routing Implementation**: React Router is used for client-side routing.

### 2.2 Backend (Node.js/Express)

* **Server Architecture**: Built using Express.js, with middleware for session management, authentication, and error handling.
* **MongoDB Database**: Used for storing user data, questions, answers, and comments.
* **API Endpoints**: RESTful API endpoints for user management, question handling, and more.
* **Authentication System**: JWT-based authentication with session management.

## 3. Features Implementation

### 3.1 User Management

* **Registration**: Users can register with their details and receive a verification email.
* **Email Verification**: Users must verify their email to activate their account.
* **Authentication**: Login and logout functionalities with session management.
* **Session Handling**: Sessions are managed using cookies and JWT tokens.

### 3.2 Core Functionality

* **Question Posting**: Users can post questions with titles, descriptions, and tags.
* **Answer Submission**: Users can submit answers to questions.
* **Commenting on Answers**: Users can comment on answers.
* **Voting Mechanism**: Users can like or dislike questions and answers.

## 

## 4. Database Schema

### User Model

**File**: User.js

**Description**: Defines the schema for user data, including fields for first name, last name, department, student ID, email, password, username, and verification status.

### Question Model

**File**: Question.js

**Description**: Defines the schema for questions, including fields for title, description, tags, user, likes, dislikes, answers, and creation date.

### Answer Model

**File**: Answer.js

**Description**: Defines the schema for answers, including fields for question ID, answer text, creation date, user, likes, dislikes, and comments.

### Comment Model

**File**: Comment.js

**Description**: Defines the schema for comments, including fields for question ID, answer ID, comment text, user, and creation date.

## 5. Security Implementation

* **JWT Authentication**: Used for secure user authentication.
* **Password Hashing**: User passwords are hashed using bcrypt before storing in the database.
* **Session Management**: Sessions are managed using express-session and stored in MongoDB.
* **CORS Configuration**: Configured to allow requests from the frontend.

## 6. User Interface Screenshots

### 6.1 Authentication Pages

**Login Page**:

A screenshot of a computer

Description automatically generated

**Registration Page**:

A screenshot of a computer

Description automatically generated

**Email Verification Page**:

A screenshot of a computer

Description automatically generated

### 6.2 Main Features

**Question Posting Interface**:

A screenshot of a computer

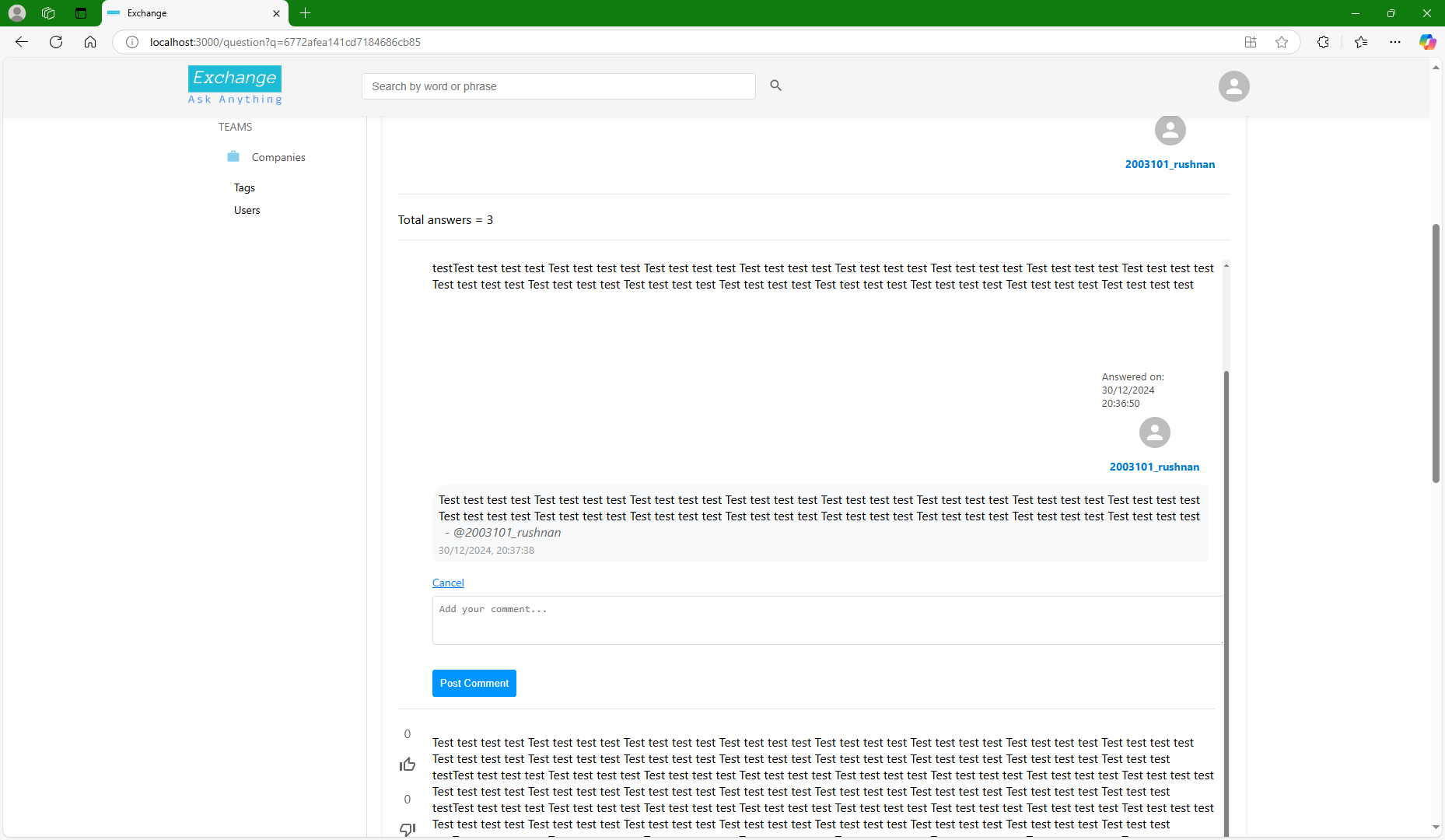
Description automatically generated

**Answer Submission Interface**:

A screenshot of a computer

Description automatically generated

**Comments Section**:



### 

### 6.3 User Profile

**Profile Page & User Statistics/Dashboard**:

A screenshot of a computer

Description automatically generated

### 6.4 Interactive Elements

**Voting Mechanism**:

A screenshot of a computer

Description automatically generated

**Search Functionality**:

A screenshot of a computer

Description automatically generated

## 7. Component Details

### 7.1 Header

* **File**: Header.js
* **Description**: Contains the navigation bar with search functionality and user menu.
* **CSS**: Header.css

### 7.2 Sidebar

* **File**: SideBar.js
* **Description**: Contains links to different sections of the application.
* **CSS**: Sidebar.css

### 7.3 Main Content

* **File**: Main.js
* **Description**: Displays the main content including questions and filters.
* **CSS**: Main.css

### 7.4 Question and Answer Components

* **Files**: Questions.js, AnswersList.js, QuestionsList.js
* **Description**: Components for displaying and interacting with questions and answers.
* **CSS**: Question.css, Allquestions.css

### 7.5 Authentication Components

* **Files**: LoginForm.js, RegisterForm.js, LoginPage.js
* **Description**: Components for user authentication.
* **CSS**: login\_page.css

### 7.6 User Profile

* **File**: UserProfile.js
* **Description**: Displays user profile information and statistics.
* **CSS**: Dashboard.css

### 7.7 Verification Components

* **Files**: Verified.js, VerErr.js, verification.js
* **Description**: Components for email verification status.
* **CSS**: Inline styles used.

### 7.8 Pagination

* **File**: pagination.js
* **Description**: Handles pagination for lists of questions and answers.

## 8. Context Providers

### 8.1 AuthContext

* **File**: authcontext.js
* **Description**: Manages authentication state and provides login/logout functions.

### 8.2 SearchContext

* **File**: searchcontext.js
* **Description**: Manages global search state.

## 9. Utility Components

### 9.1 ScrollToTop

* **File**: ScrollTop.js
* **Description**: Scrolls the page to the top on route changes.

## 10. Backend Routes

### 10.1 User Registration

* **File**: UserReg.js
* **Description**: Handles user registration, including email verification.

### 10.2 User Login

* **File**: Userlogin.js
* **Description**: Handles user login, including session management.

### User Logout

* **File**: Userlogout.js
* **Description**: Handles user logout, including session destruction.

### 10.4 Email Verification

* **File**: Verify\_email.js
* **Description**: Handles email verification for new users.

### 10.5 User Data

* **File**: userdata.js
* **Description**: Provides endpoints for fetching user profile and user-specific data.

### 10.6 Questions

* **File**: Question.js
* **Description**: Provides endpoints for creating, fetching, and interacting with questions.

### 10.7 Answers

* **File**: Answer.js
* **Description**: Provides endpoints for creating, fetching, and interacting with answers.

### 10.8 Comments

* **File**: Comment.js
* **Description**: Provides endpoints for creating, fetching, and interacting with comments.

## 11. Middleware

### 11.1 Authentication Middleware

* **File**: authenticate.js
* **Description**: Middleware for checking user authentication status.

### 11.2 Session Middleware

* **File**: session.js
* **Description**: Middleware for managing user sessions.

## 12. Utility Functions

### 12.1 Fetch User Data

* **File**: Fetchuser.js
* **Description**: Utility function for fetching user data based on session.

### 12.2 Authentication Helpers

* **File**: logauth.js
* **Description**: Helper functions for generating tokens, setting cookies, and handling logout.

## 13. Server Configuration

### 13.1 Server Setup

* **File**: server.js
* **Description**: Main server file that sets up middleware, routes, and error handling.

### 13.2 Database Connection

* **File**: db.js
* **Description**: Handles connection to MongoDB using Mongoose.

## 14. Environment Variables

* **REACT\_APP\_API\_URL**: Base URL for the backend API.
* **MONGO\_URI**: MongoDB connection string.
* **SESSION\_SECRET**: Secret key for session management.
* **JWT\_SECRET**: Secret key for JWT token generation.
* **EMAIL\_USER**: Email address for sending verification emails.
* **EMAIL\_PASS**: Password for the email account used for sending verification emails.
* **FRONTEND\_URL**: URL of the frontend application.

## 15. Folder Structure

### Backend

|  |  |
| --- | --- |
| frontend  |  ├ public  |   ├ src  ├── App.css  ├── App.js  ├── App.test.js  ├── index.css  ├── index.js  ├── logo.svg  ├── reportWebVitals.js  ├── setupTests.js  |   ├── app  | └── store.js  |   ├── components  | ├── Add-Question  | | ├── Questions.js  | | └── CSS  | | └── Question.css  | |   | ├── Body\_components  | | ├── Allquestions.js  | | ├── index.js  | | ├── Main.js  | | ├── SideBar.js  | | |   | | └── CSS  | | ├── Allquestions.css  | | ├── Index.css  | | ├── Main.css  | | └── Sidebar.css  | ├── dashboard  | | ├── AnswersList.js  | | ├── QuestionList.js  | | ├── Dashboard.js  | | ├── pagination.js  | | ├── UserProfile.js  | | └── Dashboard.css  | |   | ├── Header  | | ├── Header.js | | | ├── logo-no-background.png  | | ├── logo\_no\_background\_dark.png  | | |   | | └── CSS  | | └── Header.css  | |   | ├── Login\_page  | | ├── LoginFrom.js  | | ├── login\_page.js  | | ├── RegisterForm.js  | | |   | | └── CSS  | | └── login\_page.css  | |   | ├── Protected  | | └── Protected.js  | |   | ├── ScrollTop  | | └── ScrollTop.js   | |   | ├── Verified  | | ├── VerErr.js  | | ├── verification.js  | | └── Verified.js  | |   | └── ViewQuestion  | ├── Index.js  | ├── MainQue.js  | └── CSS  | └── Index.css  |   ├── context  | ├── authcontext.js  | └── searchcontext.js  |   ├── dev  |   └── features  └── counter |

### Frontend

|  |
| --- |
| ├── README.md ├── exchange/  ├── backend/  ├── middleware/  │ ├── authenticate.js  │ └── session.js  ├── models/  │ ├── Answer.js  │ ├── Comment.js  │ ├── Question.js  │ └── User.js  ├── routes/  │ ├── Answer.js  │ ├── Comment.js  │ ├── Index.js  │ ├── Question.js  │ ├── userdata.js  │ ├── Userlogin.js  │ ├── Userlogout.js  │ ├── UserReg.js  │ └── Verify\_email.js  ├── helper/  │ ├── Fetchuser.js  │ └── logauth.js  ├── db.js  └── server.js |

.