SSN COLLEGE OF ENGINEERING

COMPUTER SCIENCE AND ENGINEERING

UCS2611 INTERNET PROGRAMMING LAB

MINI PROJECT (QUIZ API)

TEAM MEMBERS:

G Vaishal Raj 3122215001119

Sudharsan 3122215001111

Date :11-05-2024

DESCRIPTION:

Full stack web application for conducting *On-line quiz using MVC architecture*. The application facilitates the normal and admin users to access it. To the normal user, instructions, questions with options and the score is provided. Admin user can view the registered users and their scores.

Index.js

This React code sets up routing for a web application. It uses React Router for navigation. When users access different URLs, the application renders different components accordingly. The main components involved are `App`, `Quiz`, and `Score`.

- The `BrowserRouter` component wraps the application and provides routing capabilities.

- Inside it, `Routes` component is used to define different routes.

- Each `Route` component specifies a path and the corresponding component to render.

- The root path ("/") renders the `App` component.

- Paths starting with "/Quiz/:username/:email" render the `Quiz` component, with dynamic parameters for username and email.

- Paths with an additional score parameter ("/Quiz/:username/:email/:score") render the `Score` component, which presumably displays the quiz score.

Overall, this code organizes the application's structure and navigation flow, allowing users to interact with different components based on the URL they access.

Server.js

This is a Node.js Express server that serves as the backend for a quiz application. It connects to a MongoDB database using Mongoose for data storage. The server provides several endpoints:

1. User Authentication:

- `/getUser`: Validates user login credentials (email and password) by querying the MongoDB database for a matching user entry. If found, it returns a message indicating successful validation; otherwise, it returns an error message.

- `/addUser`: Adds a new user to the database after validating that the email doesn't already exist.

2. **Quiz Functionality:**

- `/getQs`: Retrieves quiz questions from the database and sends them as JSON to the client.

- `/updateScore`: Updates the score of a user in the database based on their email.

The server also handles CORS and JSON parsing middleware. It listens on port 3001 for incoming connections.

App.js:

App.js serves as the entry point of the React application. It's responsible for setting up routing using the BrowserRouter component from react-router-dom. This allows the application to have multiple views based on the URL. Inside the <Routes> component, three routes are defined. The first route, '/', renders the Login component, where users authenticate. The second route, '/users', renders the People component, displaying users and their scores. The third route, '/quiz', renders the Quiz component, where users take the quiz. Additionally, App.js initializes the users state variable using the useState hook, which is crucial for managing user data across the application.

Login.js

Date :11-05-2024

This is a React component called `Login` which handles user authentication. It includes form inputs for username, email, and password. Users can either sign in or sign up using the provided buttons. The component uses state to manage form inputs, visibility of the password field, and authentication feedback. Upon submission, it sends a POST request to the backend server (`http://localhost:3001`) with user credentials. If the user is authenticated successfully, it redirects to the quiz page with the user's username and email. If there's an error or authentication fails, appropriate feedback is displayed to the user.

Admin.js

This React component serves as an admin dashboard displaying user information. It fetches data from a server endpoint that retrieves details of all users, including their usernames, emails, and scores. The retrieved data is then mapped to generate a table where each row corresponds to a user entry, showcasing their respective details. The dashboard provides an organized view for the admin to monitor and manage user information efficiently.

Quiz.js

This React component represents a quiz-taking interface. It fetches questions from a local server and displays them one at a time. Users select an answer within a time limit, with the timer displayed. After selecting an answer, the component checks if it's correct and updates the score accordingly. When all questions are answered or the timer runs out, the quiz is submitted. Upon completion, it updates the user's score on the server and redirects to a summary page displaying the user's username, email, and final score.

Score.js

This React component represents a score summary page displayed after completing a quiz. It dynamically retrieves the username, email, and score from the URL parameters using the `useParams` hook from React Router. The user's username and score are prominently displayed in the center of the page. Additionally, it provides a button to redirect the user back to the login page using a `Link` component from React Router.

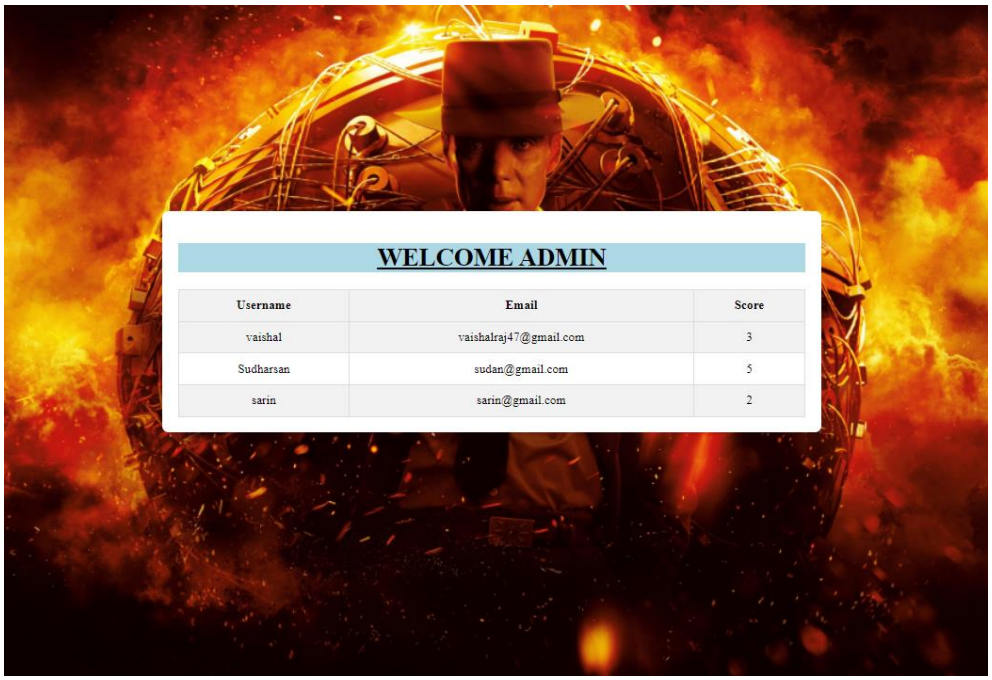
Website Flow:

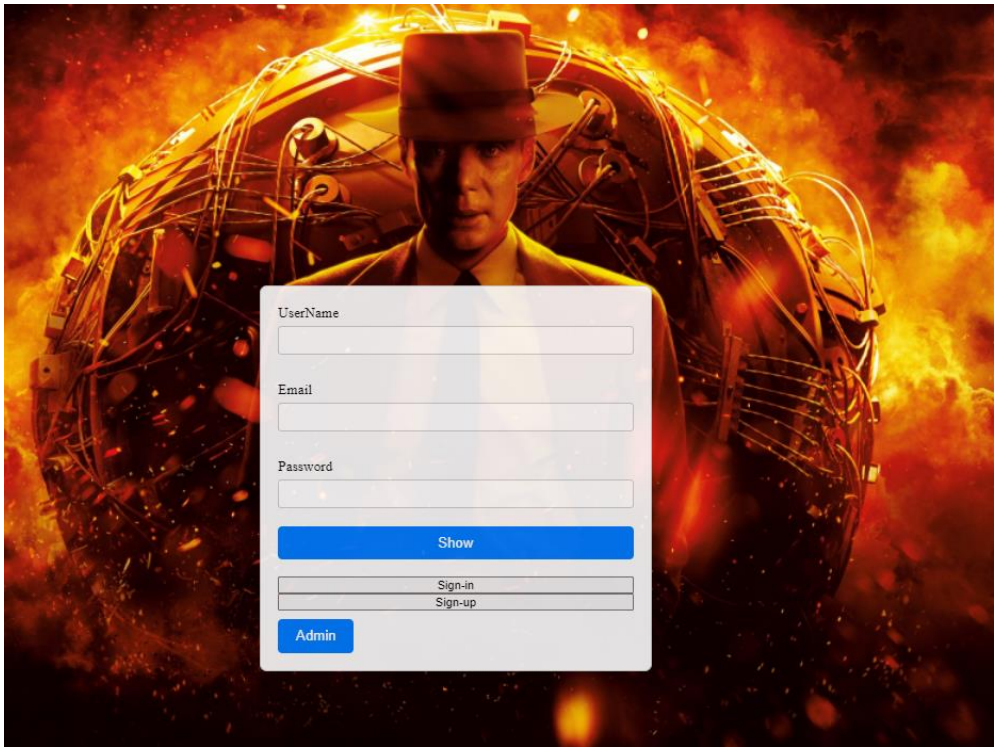
Date :11-05-2024

When a user visits the website, they are directed to the login page.

This page checks the authenticity of the users. Provided a simple button for Admin.

Admin----------

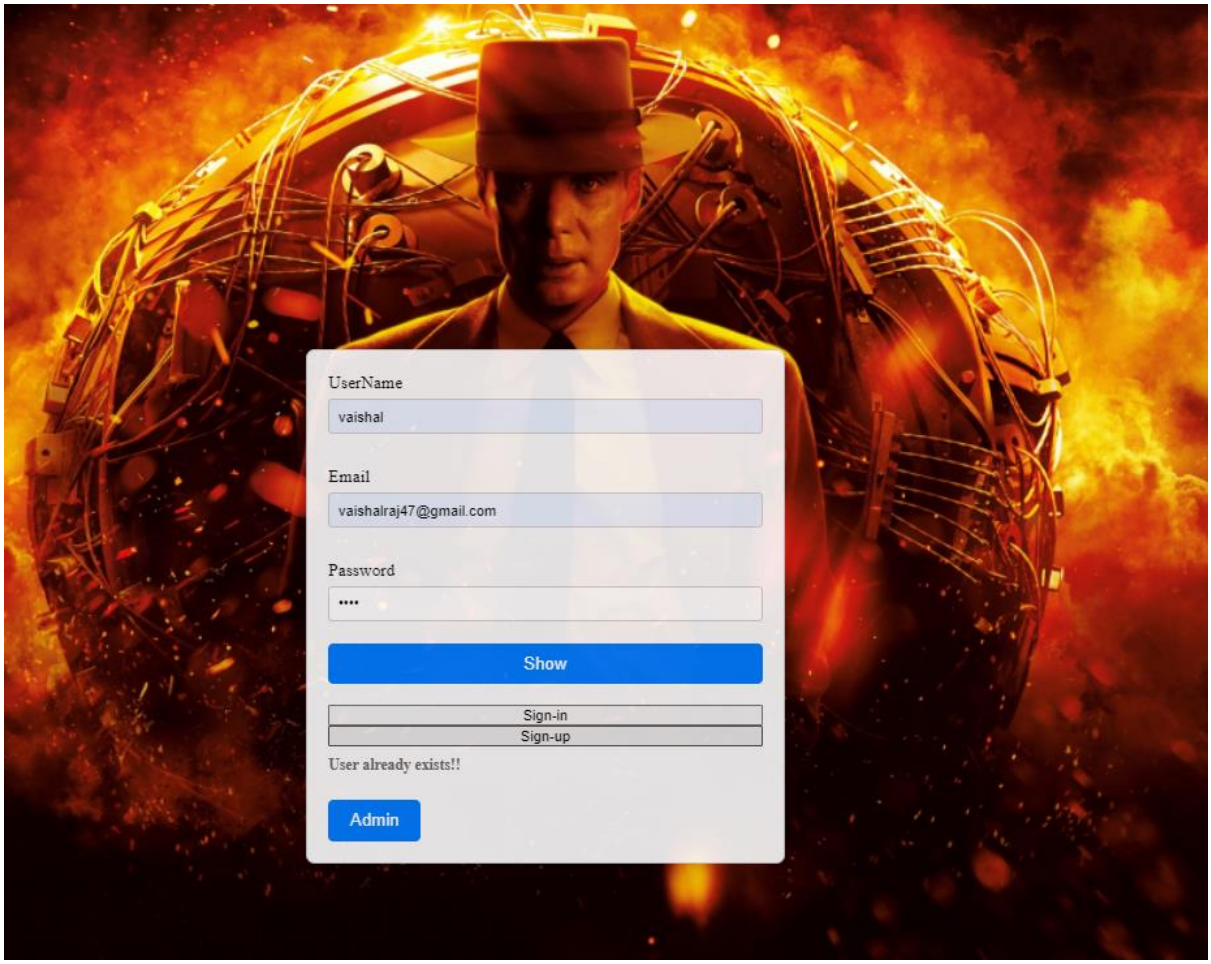




Date :11-05-2024

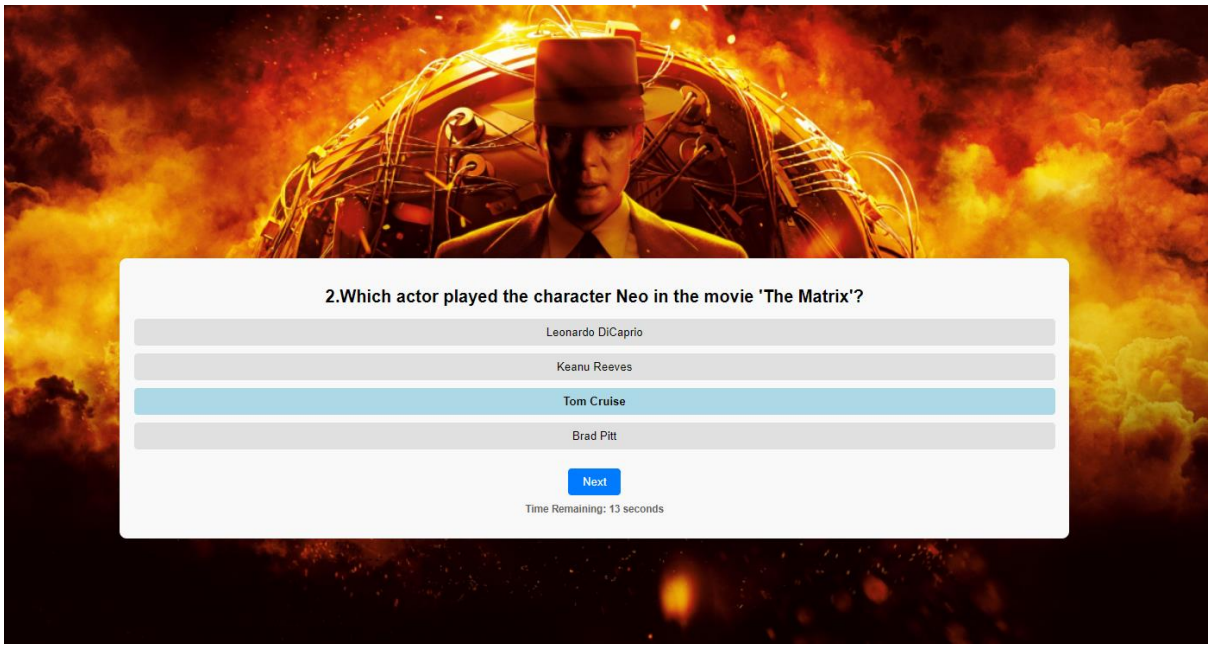
We have provided both sign-in and register options.

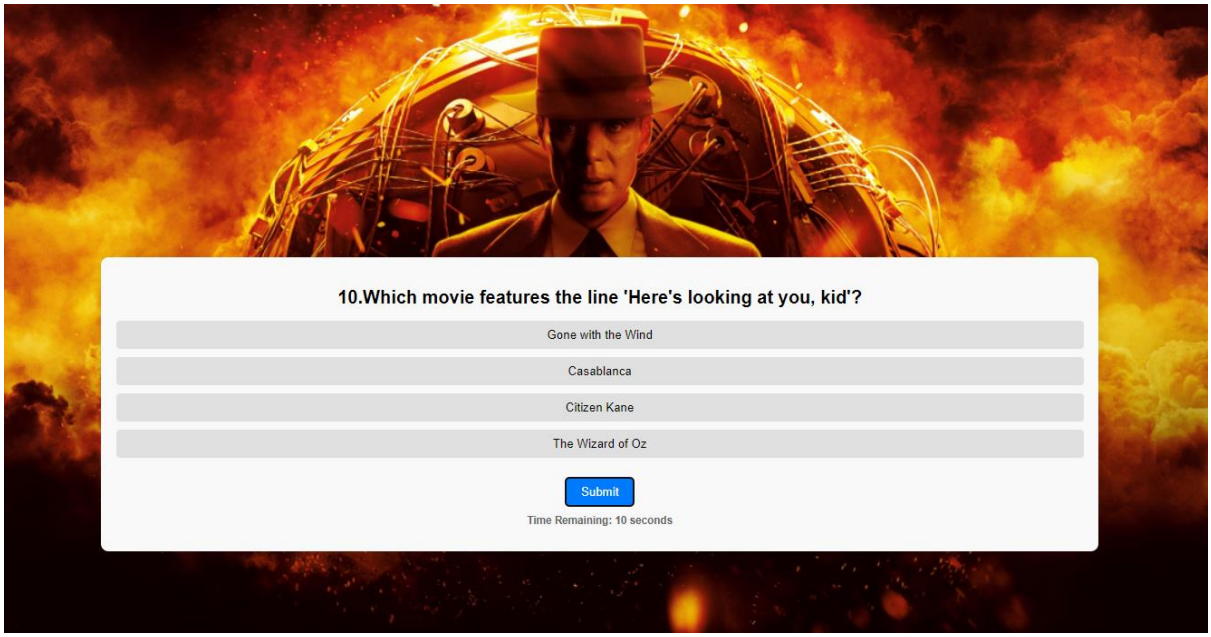
If user clicks register it first checks whether the user already exists or not.



Date :11-05-2024

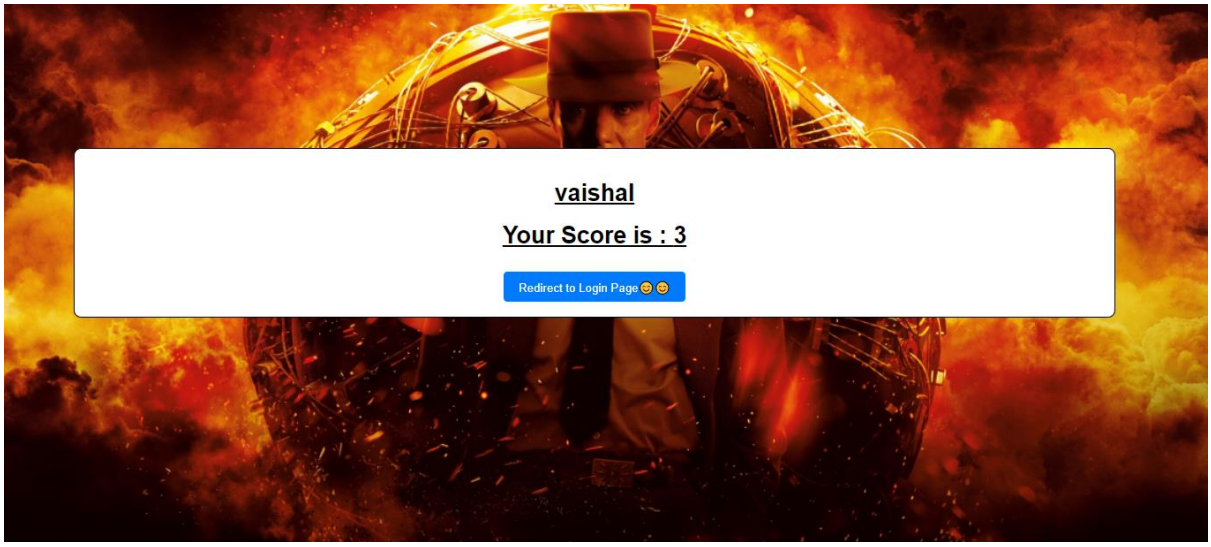
Then user takes the quiz.





After completing the quiz, users are redirected to the score page, where they can view their score.

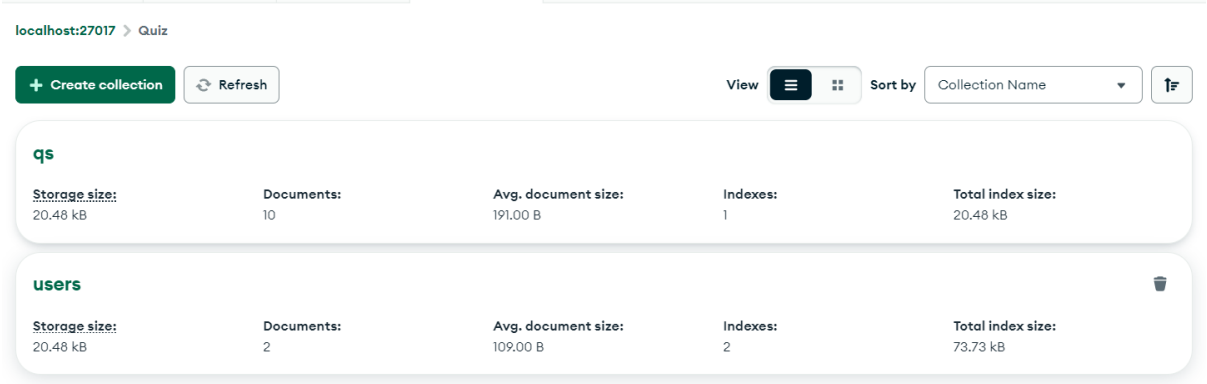
Date :11-05-2024

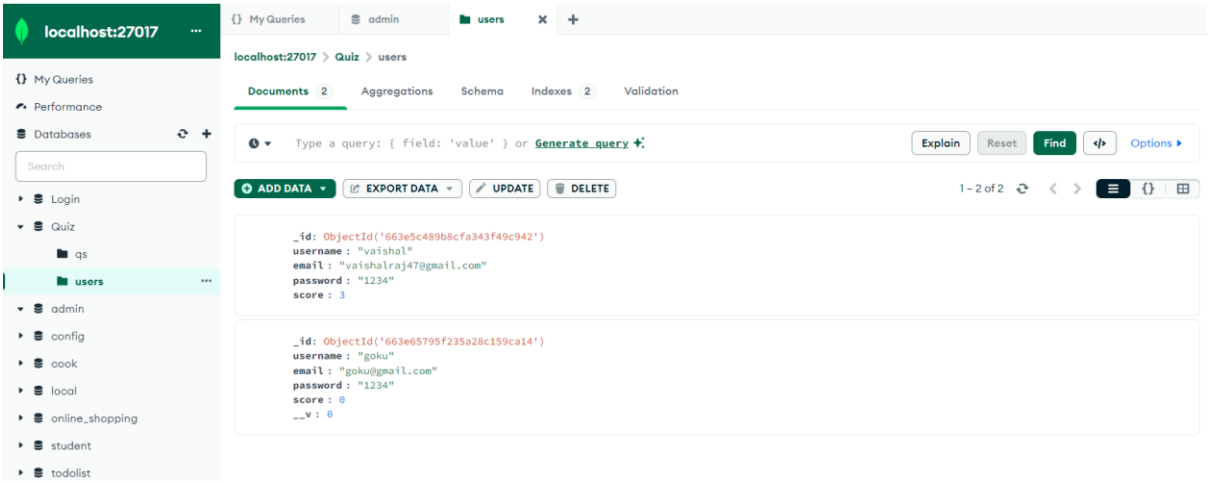


From the score page or users' page, users can navigate back to the login page if needed.

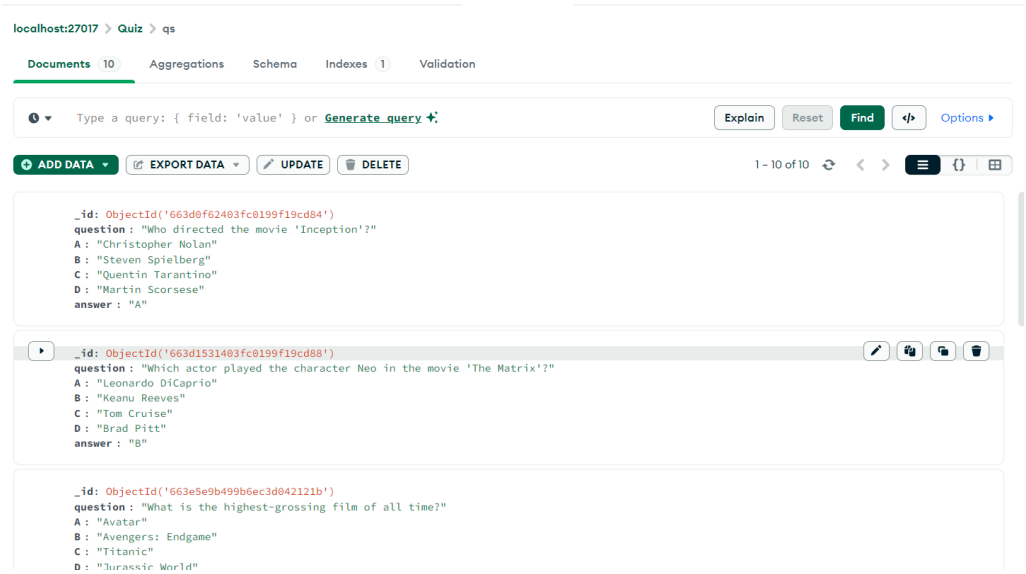
Date :11-05-2024

SCHEMA:





Questions:



Date :11-05-2024

Users:


```
_id: ObjectId('663e5c489b8cfa343f49c942')
username : "vaishal"
email : "vaishalraj47@gmail.com"
password : "1234"
score : 3
```

```
_id: ObjectId('663e65795f235a28c159ca14')
username : "Sudharsan"
email : "sudan@gmail.com"
password : "1234"
score : 5
__v : 0
```

```
_id: ObjectId('663faf012fe4b58db9cd64fd')
username : "sarin"
email : "sarin@gmail.com"
password : "1234"
score : 2
```