

# Zense club recruitment project

Abhinav Deshpande [See the project](#)

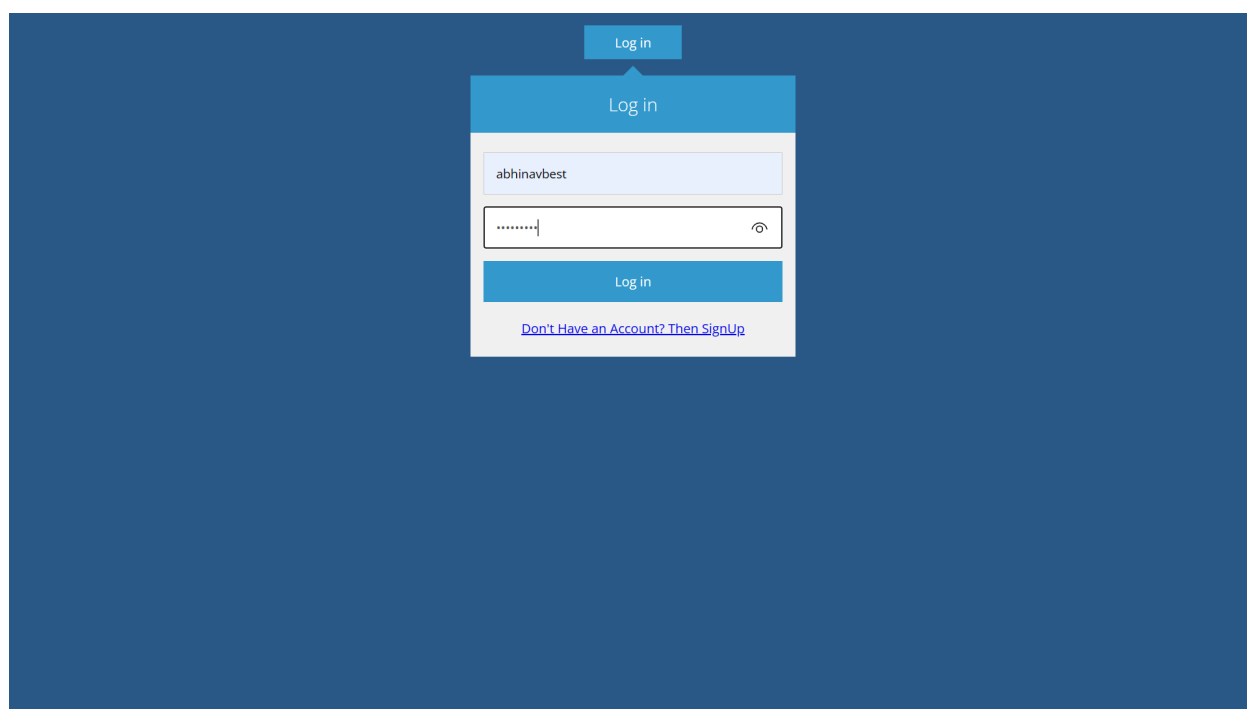
## Quizzer

Please refer to readme.md to start with this project. You can view it [here](#).

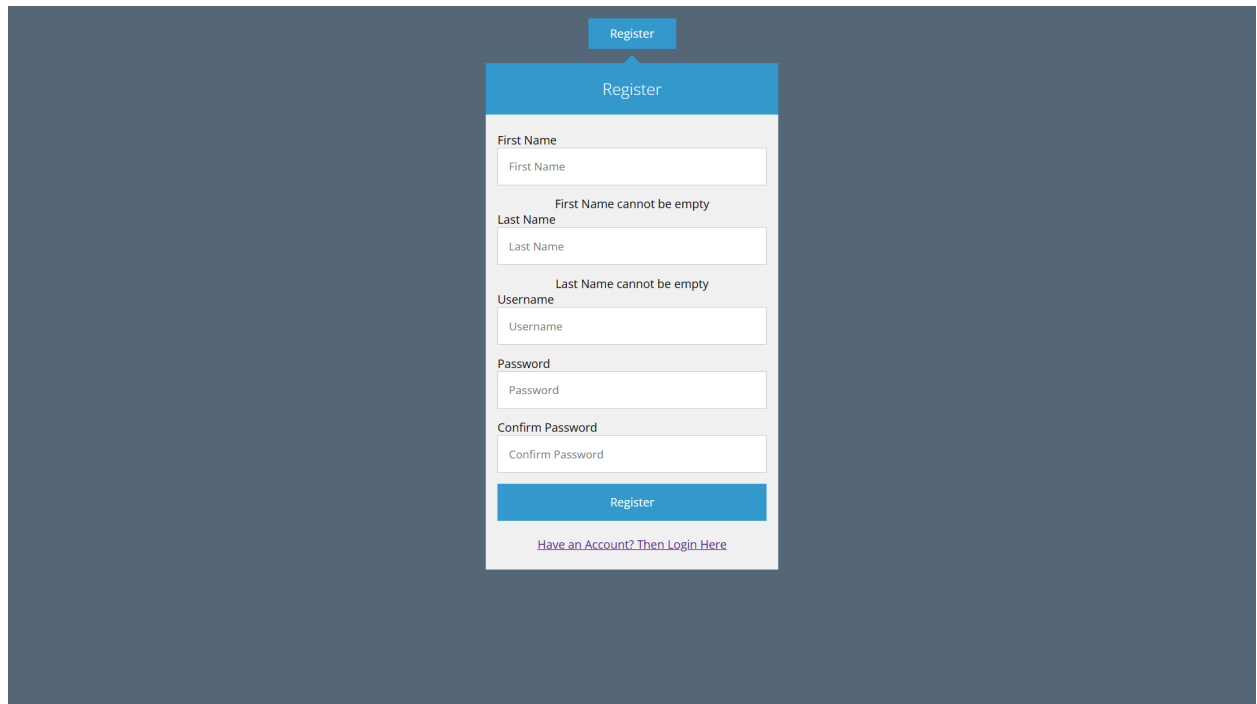
### Project Demo

To run locally, please do so by following the steps in readme.md file. Here are some screenshots of the application.

The login page

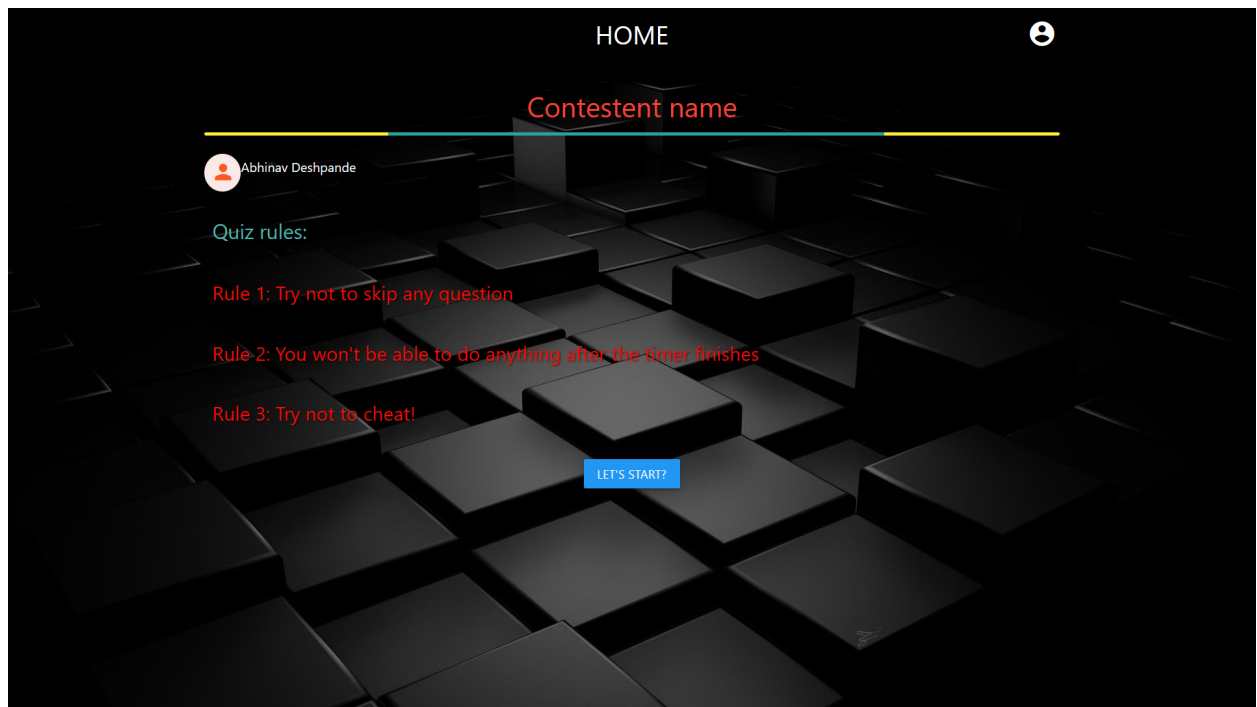


Don't have an account? Then register here!

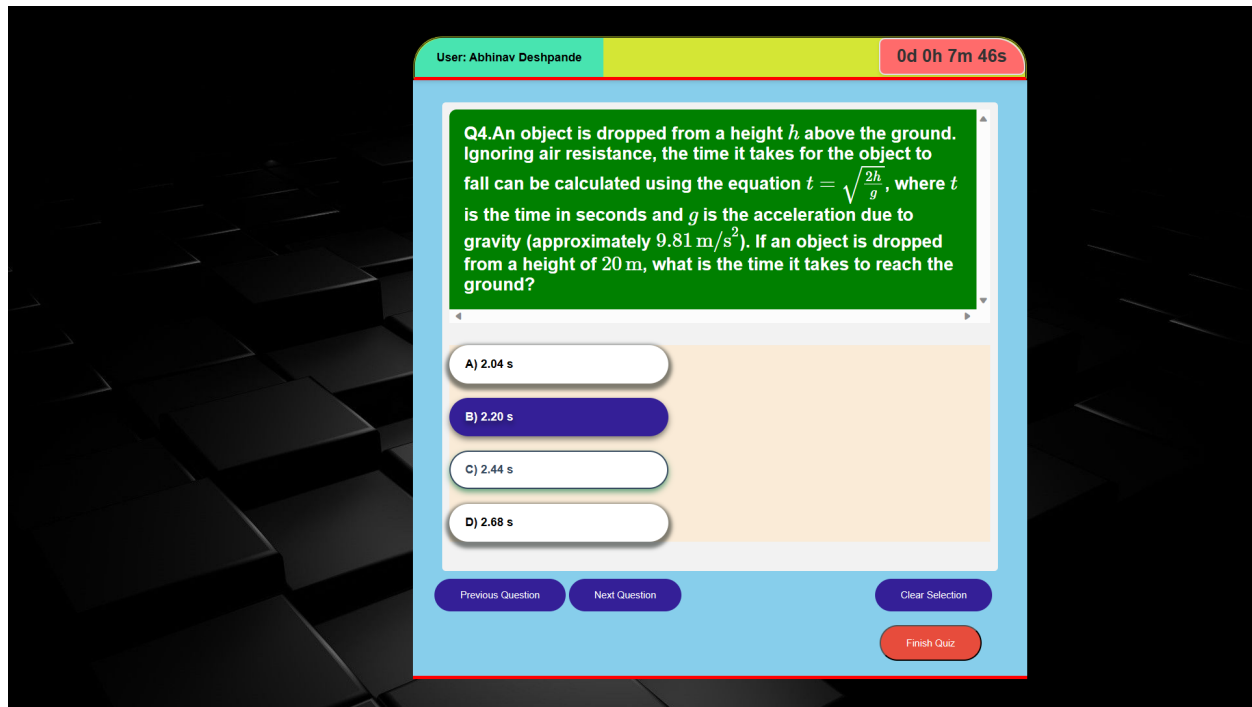


A screenshot of a web application's registration form. The form is centered on a dark blue background. At the top, there is a blue button labeled "Register". Below it, the form has a blue header with the word "Register" in white. The form contains several input fields: "First Name", "Last Name", "Username", "Password", and "Confirm Password". Each input field has a corresponding label above it. Below the "First Name" and "Last Name" fields, there are error messages: "First Name cannot be empty" and "Last Name cannot be empty". At the bottom of the form, there is a blue button labeled "Register" and a link that says "Have an Account? Then Login Here".

Home page:

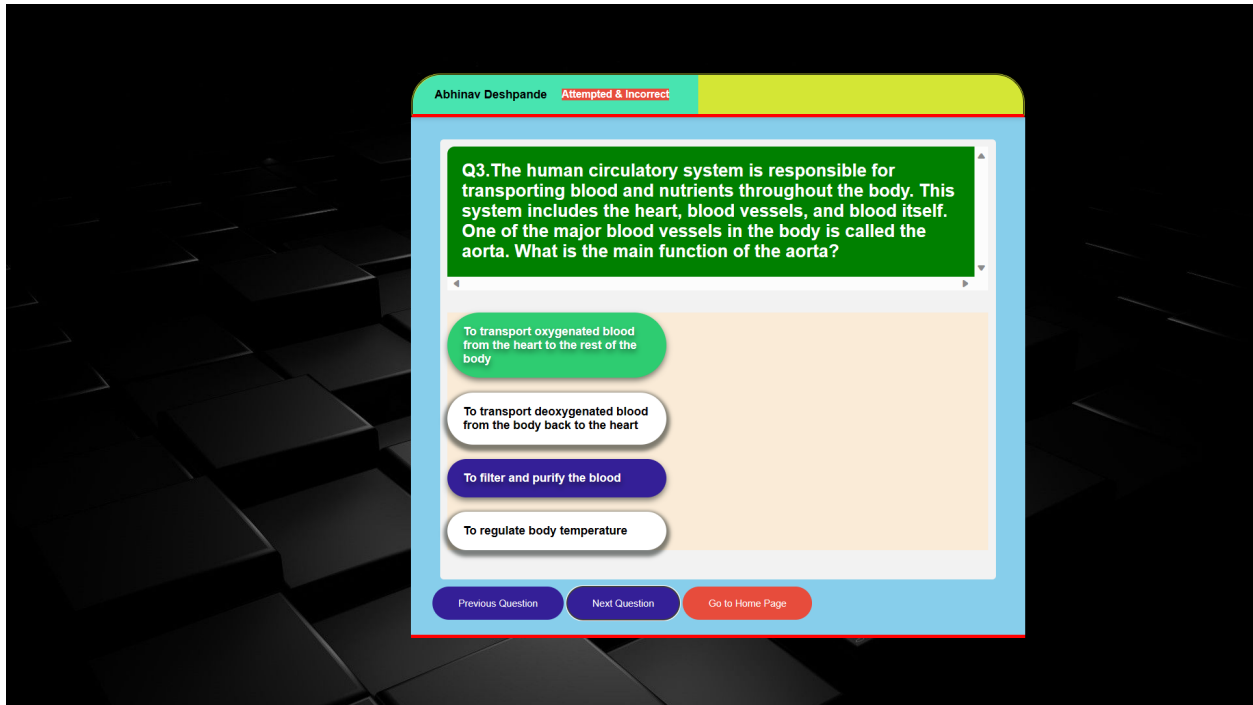


## The quiz



## Quiz completion

## The quiz analysis



## Features | Working explained | Tech used

### Authentication and Authorization:

- Middleware for authentication and authorization ensures that users can access specific routes only when logged in.
- Users need to log in to access the quiz, analysis, and home pages.

### Routing and Serving Static Files:

- Express routing is used to define routes for different pages of the application.
- Static files like HTML, CSS, and JavaScript are served from the "public" directory.

### Session Management:

- Express-session middleware manages user sessions and stores session data.

#### Database Integration:

- MySQL database is used for storing user data, quiz questions, and user quiz data.
- mysql2 is used as a database driver.
- Database connection is established, and queries are executed to interact with the database.

#### Quiz Functionality:

- Users who are authenticated can access the quiz page.
- Quiz questions are fetched from the database and displayed to the user.
- Users can select and store their answers for each question.
- User quiz data is stored in the user\_quiz\_data table in the database.

#### Analysis Page:

- Authenticated users can access the analysis page.
- Users can view their quiz performance, including marked answers and correct answer and attempt status for each question.

#### Error Handling:

- 404 and 500 error handling middleware are implemented to handle page not found and server error scenarios.

#### User Interaction:

- Users can register, log in, and take quizzes.
- Quiz data is stored, and users can see their quiz performance.

#### Modularization:

- The project is structured into separate modules for routing, middleware, database interaction, and more.

Start Server:

- The application listens on a specified port (default is 3000) and starts the server.
- Console log messages indicate the status of the server.

Overall, this project combines Express.js, MySQL, and various middleware to create a functional quiz application where users can register, log in, take quizzes, and view their performance analysis. The application demonstrates integration between a web server, a database, and user authentication, making it a comprehensive example of a web application. HTML and CSS are used for designing the front end.

## Future Scope

As an open-source project, it is expected that there will be contributions by many people in the future. Please refer to the README.md file [here](#) for more on this topic.