

# Project Documentation

## 1. Online Practice Test Platform

### Overview

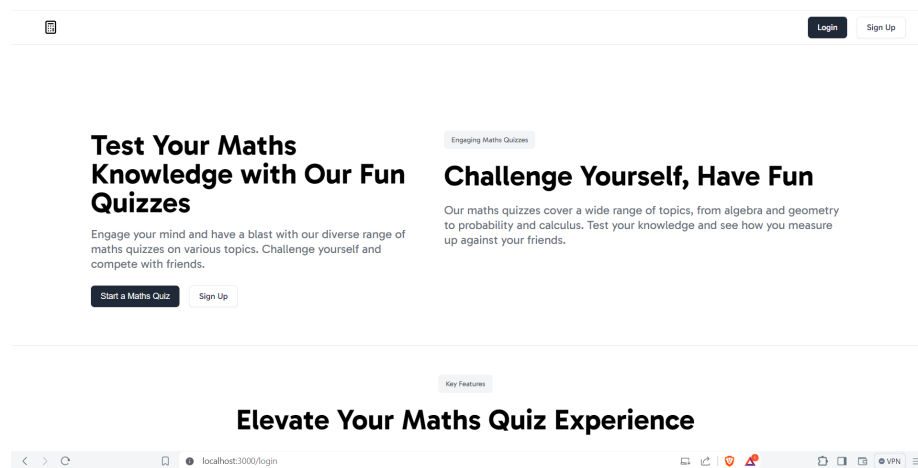
This project is a quiz application that allows users to register, log in, and take quizzes on various topics. The application features user authentication, dynamic quiz generation, and personalized quiz statistics.

### Video

### Project Demo

<https://github.com/Nihar4/Assignment/assets/91326796/68886256-82d9-402a-af4a-7f24b236cf6b>

### Screenshot



Welcome back

Email

me@example.com

Password

Forgot password?

Sign in

Sign in with Google

Don't have an account? Sign up


**Create an account**

**Name**

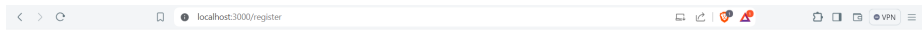
**Email**

**Password**

**Sign up**


 Sign in with Google

Already have an account? [Sign in](#)














Sign in - Google Accounts - Brave

accounts.google.com/gsi/select?client\_id=353327592409-fm...

 Sign in with Google

**Choose an Account**  
to continue to test-google-drive

-  
-  
-  
-  
-  

 Use another account

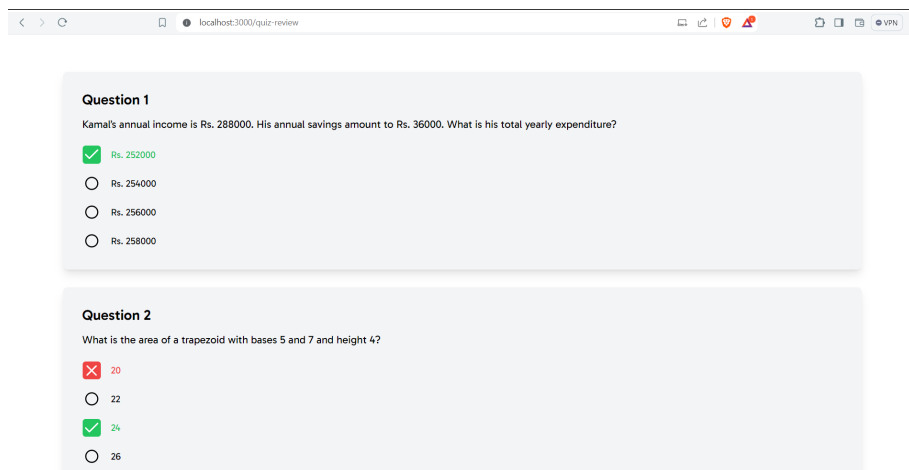
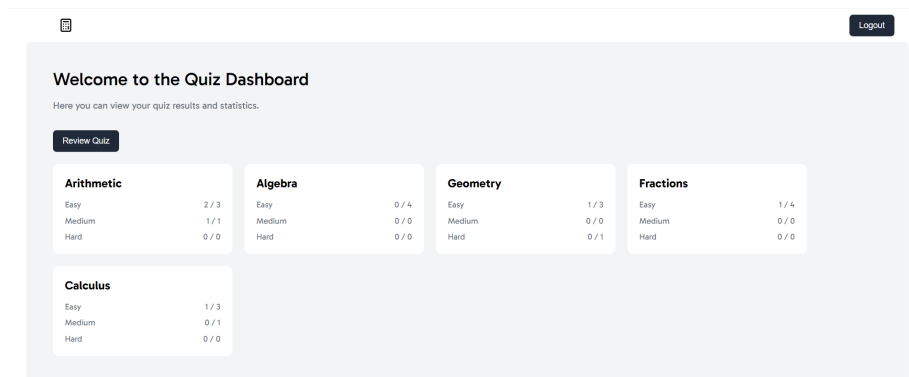


**Question 4**

Solve for  $x$ :  $x - 4 = 10$

- ☐ 13
- ☒ 14
- ☐ 15
- ☐ 16

Next



## Setup Instructions

### 1. Clone the Repository:

```
git clone https://github.com/Nihar4/Assignment.git
cd Assignment/Online_Practice_Test_Platform
```

### 2. Install Dependencies:

- Backend:

```
cd backend
npm install
```

- Frontend:

```
cd ../frontend
npm install
```

### 3. Set Up Environment Variables:

- Create a `.env` file in the backend directory and add the following:

```
PORT=5000
MONGO_URI=
JWT_SECRET=
FRONTEND_URL =
GOOGLE_CLIENT_ID =
GOOGLE_CLIENT_SECRET =
GOOGLE_CALLBACK_URL =
SESSION_SECRET =
```

#### 4. Run the Application:

- Start the backend server:

```
cd backend
npm start
```

- Start the frontend development server:

```
cd ../frontend
npm start
```

5. **Access the Application:** Open your browser and navigate to `http://localhost:3000`.

## Approach and Assumptions

### Approach

- **Backend:** The backend is built using Express.js and MongoDB. It includes user authentication, quiz management, and question handling.
- **Frontend:** The frontend is built using React.js. It includes pages for registration, login, home, quiz, dashboard, and quiz review.
- **Authentication:** The application uses JWT for user authentication and authorization.
- **Quiz Logic:** Quizzes are dynamically generated based on predefined tags and difficulty levels. The backend tracks user responses and adjusts the difficulty of subsequent questions accordingly.

### Assumptions

- Each quiz consists of a predefined number of questions (e.g., 20 questions).
- Users can only take one quiz at a time.
- Tags are predefined and cannot be added dynamically through the UI.
- The system uses a basic algorithm to adjust question difficulty based on the user's performance.

## 2. Web Scraping Script

### Overview

This Python script scrapes property details from the real estate website realestate.com.au for a specific location. It extracts information such as property address, type, price, bedrooms, bathrooms, parking spaces, square footage, auction details, and agent details.

### Setup Instructions

#### Prerequisites

- Python latest version
- Required Python libraries: `requests`, `bs4`, `csv`, `json`

#### Installation

1. Clone the repository:

```
git clone https://github.com/Nihar4/Assignment
```

2. Navigate to the project directory:

```
cd Web_Scraping_Script
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Run the script:

```
python Web_Scraping.ipynb
```

### Approach

The script follows these main steps:

1. **Fetching and Parsing Data:** The `fetch_and_parse` function is used to fetch HTML data from the specified URL using the `requests` library. It then parses the HTML using `BeautifulSoup` to extract property details.
2. **Extracting Property Details:** The `extract_property_details` function extracts various property details from the parsed HTML data using `BeautifulSoup`. It searches for specific HTML elements containing property information such as address, type, price, bedrooms, bathrooms, etc.
3. **Saving Data:** The extracted property details are stored in a list of dictionaries. The data is then saved to CSV and JSON files using the `csv` and `json` libraries, respectively.
4. **Pagination:** The script iterates through multiple pages of property listings by following the pagination links until there are no more pages left.

## Assumptions

- The script assumes that property details are consistently structured across different listings on the real estate website. Any changes to the website's HTML structure may require adjustments to the parsing logic.
- It assumes that all required information (e.g., address, type, price) is present on each property listing page. If certain details are missing, they will be omitted from the output.
- Pagination is assumed to be available and follows a standard format of "Go to Next Page" links. If the website's pagination structure changes, the script may need to be modified accordingly.

## Json Demo

property\_details - Copy (2).json

## CSV Demo

property\_details - Copy (2).csv