

Mock Test Generator — Project Documentation

Project Overview

Mock Test Generator is a responsive and intelligent web application that allows users to create customized mock tests using React and AI APIs (Gemini/OpenAI) or any other ai via api. It offers an intuitive interface for test-taking, real-time evaluation, and insightful performance analytics.

Features

Custom Test Configuration

- Select one or more topics (e.g., React, DSA, Math, etc.)
- Choose difficulty level: Easy, Medium, Hard
- Choose question type: Single correct / Multiple correct
- Enable/disable negative marking
- Set number of questions and test duration

AI-Powered MCQ Generation

- Uses Gemini or OpenAI API or any other api key to dynamically generate relevant MCQs
- Supports math symbols and LaTeX via MathJax

Real-Time Test Environment

- Clean UI with a timer that supports pause/resume
- Navigation between questions
- Auto-submission on time expiration

Performance Result Dashboard

- Displays total score and percentage
- Topic-wise accuracy with color-coded indicators
- Performance classification: Excellent, Good, Needs Practice
- Full review panel showing correct/wrong answers with highlights

Tech Stack

Layer	Tools Used
Frontend	React, Bootstrap 5, MathJax
Routing	React Router DOM

Styling	Bootstrap 5 + Custom CSS
API Layer	Gemini / OpenAI/any other ai api key (via generateMCQs() helper)
State Mgmt	React Hooks (useState, useEffect)

Setup Instructions

1. Step 1: Clone the Repository

```
git clone https://github.com/GauravAttri422/mock-test-generator.git
```

```
cd mock-test-generator
```

2. Step 2: Install all Dependencies

```
npm install
```

3. Step 3: Add Your API Key

Create a file named .env in the project root and add:

```
REACT_APP_GOOGLE_API_KEY=your_gemini_or_chatgpt_api_key
```

4. Step 4: Start the Application

```
npm start
```

Visit: <http://localhost:3000>

Folder Structure

```
mock-test-generator/
├── public/
│   └── index.html
└── src/
    ├── components/
    ├── pages/
    ├── utils/
    ├── styles/
    └── App.jsx
├── .env
└── README.md
```

License

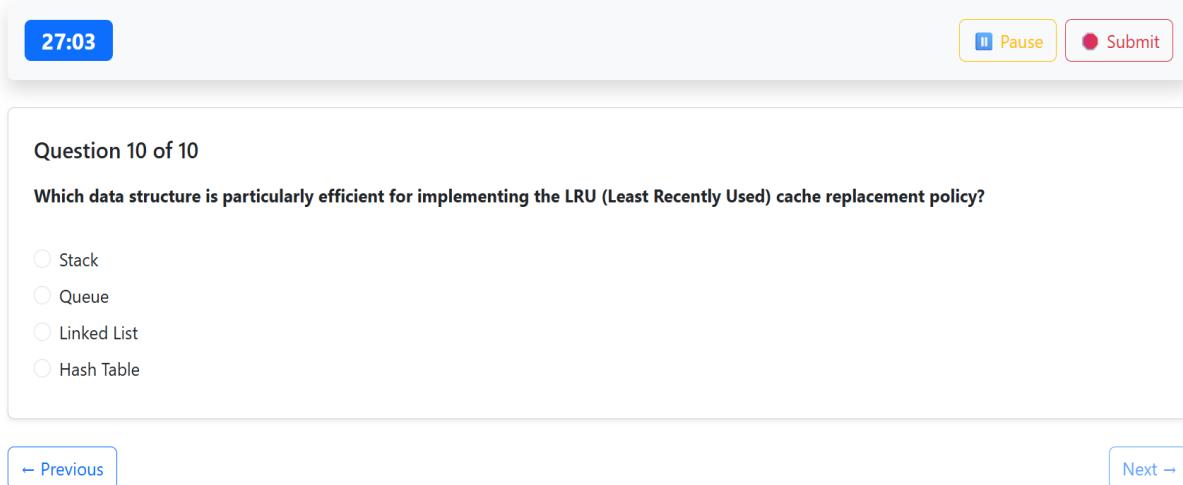
This project is released under the MIT License — free to use, modify, and share.

Author

Gaurav Attri

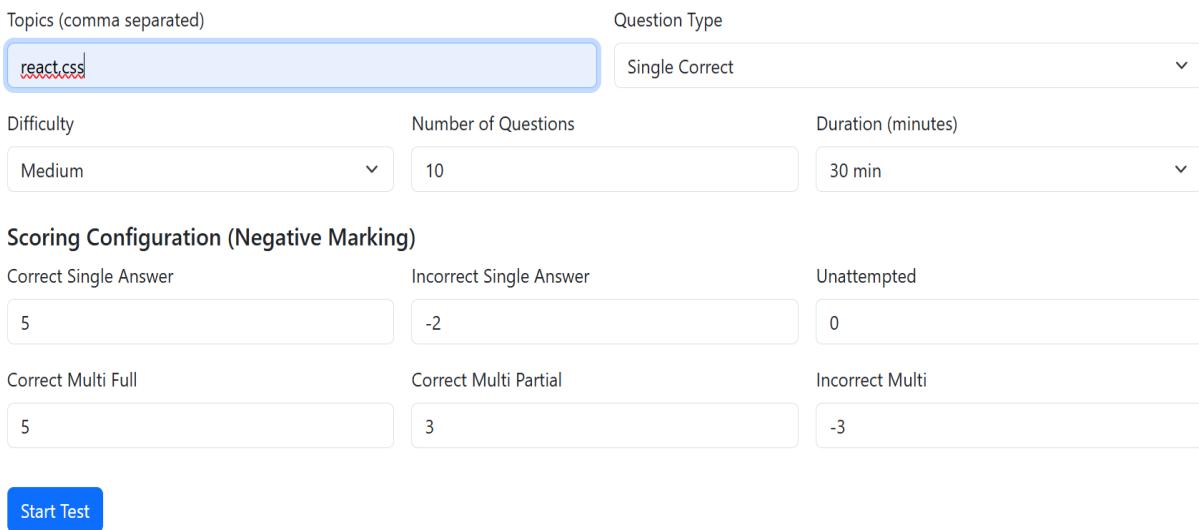
GitHub: <https://github.com/Gauravattri422>

Screenshots of the projects



The screenshot shows a mock test interface. At the top right are 'Pause' and 'Submit' buttons. On the left, a timer displays '27:03'. The main area is titled 'Question 10 of 10' and asks: 'Which data structure is particularly efficient for implementing the LRU (Least Recently Used) cache replacement policy?'. Below the question are four options: 'Stack', 'Queue', 'Linked List', and 'Hash Table', each preceded by an empty radio button. At the bottom are navigation buttons: '← Previous' on the left and 'Next →' on the right.

Create Your Mock Test



The screenshot shows the configuration interface for creating a mock test. It includes fields for 'Topics (comma separated)' containing 'react,css', 'Question Type' set to 'Single Correct', 'Difficulty' set to 'Medium', 'Number of Questions' set to '10', and 'Duration (minutes)' set to '30 min'. Below this is a 'Scoring Configuration (Negative Marking)' section with six input fields arranged in a 2x3 grid. The first row contains 'Correct Single Answer' (5), 'Incorrect Single Answer' (-2), and 'Unattempted' (0). The second row contains 'Correct Multi Full' (5), 'Correct Multi Partial' (3), and 'Incorrect Multi' (-3). At the bottom is a blue 'Start Test' button.

Correct Single Answer	Incorrect Single Answer	Unattempted
5	-2	0

Correct Multi Full	Correct Multi Partial	Incorrect Multi
5	3	-3

Start Test

Topic-wise Accuracy

CSS

40.0%

Review Question 1 of 10

Which CSS property controls the space between the content of an element and its border?

padding

margin

border-spacing

width

← Previous

Next →

 Retake Test

Test Results

Overall Score

12 out of 50 correct (24.00%)

Quiz Analytics Dashboard

Overall Performance

50.0% Correct

Keep practicing!

Correct

4

Incorrect

4

Unattempted

2