

Trivia Trek

Introduction:-

Project Description

Trivia Trek is an interactive quiz application designed to make learning engaging and accessible for users of all age groups. It provides multiple-choice questions across a wide range of topics and allows users to track their performance through scores, timers, and detailed result summaries. Users can review incorrect answers, understand explanations, and continuously improve their knowledge. The application focuses on delivering an educational yet enjoyable experience through a simple and intuitive interface.

Introduction

Trivia Trek introduces a modern approach to learning through interactive quizzes. It is built with a user-centric design that combines simplicity with efficient functionality. The platform provides topic-wise quizzes, instant scoring, time-bound challenges, and performance analytics to enhance the overall learning process. Trivia Trek aims to make knowledge exploration more engaging by transforming traditional learning into an interactive experience.

Core System Overview

Trivia Trek is designed with a responsive and efficient architecture that ensures smooth performance across devices. Each quiz presents a set of multiple-choice questions along with a timer, scoring logic, and clear feedback for every attempt. At the end of each quiz, users receive a detailed summary of their performance, including the total score, correct and incorrect responses, and explanations for the questions they answered incorrectly.

The application emphasizes ease of use, clear navigation, and quick interaction, making it suitable for both quick practice sessions and extended learning.

Scenario-Based Introduction

Consider a student preparing for competitive exams or simply wanting to test their general knowledge. They open Trivia Trek and are greeted with a clean and simple interface displaying various quiz categories.

The student selects a category, such as Science and Technology. A timer starts, and the quiz begins. They answer each question while keeping track of time and score. After completing the quiz, the student receives a detailed results page that includes:

- Total number of questions attempted
- Correct and incorrect responses
- Time taken
- Explanations for incorrect answers
- Suggestions for improvement

This process encourages the student to revisit topics, retake quizzes, and strengthen their understanding through active practice.

Target Audience

Trivia Trek is designed for a broad user base, including:

Students: Individuals preparing for examinations or wanting to strengthen their knowledge across subjects.

Knowledge Enthusiasts: Users who enjoy testing and expanding their general awareness.

Educators: Teachers who want to conduct quick assessments or introduce interactive learning.

General Users: Anyone seeking to learn or engage in quizzes for entertainment and self-improvement.

Project Goals and Objectives

The primary aim of Trivia Trek is to provide an accessible and engaging quiz-based learning platform. Its objectives include:

User-Friendly Interface: Develop a clean and intuitive interface that enables smooth navigation, easy quiz selection, and clear presentation of results.

Effective Quiz Management: Incorporate structured quizzes, timers, scoring systems, and a detailed results summary to support learning and progress tracking.

Learning Enhancement: Provide explanations for incorrect answers to help users understand concepts and improve knowledge retention.

Engaging User Experience: Use interactive elements such as timed questions and performance analytics to maintain user motivation.

Scalable Modern Technology Stack: Build the application using modern web technologies to ensure fast loading, responsive design, and scalability for future features such as leaderboards, user accounts, and advanced analytics.

Key Features:-

➤ **Interactive Quiz System:**

A real-time quiz interface that displays multiple-choice questions, tracks the timer, and updates scores instantly as the user progresses.

➤ **Category-Based Quiz Modules:**

Separate quiz sections for different topics such as General Knowledge, Science, Technology, History, and more, allowing users to choose and attempt quizzes based on their interests.

➤ **Performance Tracking and Review:**

A detailed performance summary is provided at the end of each quiz, including correct answers, incorrect answers, explanations, total score, and time taken.

➤ **Responsive and Cross-Device Compatibility:**

The application is designed to function smoothly on different devices such as mobiles, tablets, and desktops, ensuring a consistent and accessible quiz experience.

PRE-REQUISITES:-

Here are the key prerequisites for developing a frontend application using React.js:

➤ **Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the local environment. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

- Download: <https://nodejs.org/en/download/>

- Installation instructions: <https://nodejs.org/en/download/package-manager/>

➤ **React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

- Create a new React app:

```
npm create vite@latest
```

Enter and then type project-name and select preferred frameworks and then enter

- Navigate to the project directory:

```
cd project-name
```

```
npm install
```

- Running the React App:

With the React app created, you can now start the development server and see your React application in action.

- Start the development server:

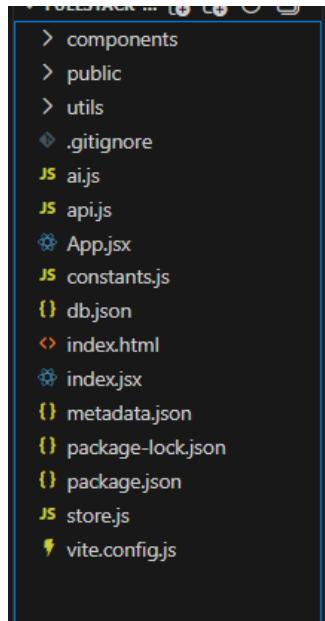
```
npm run dev
```

This command launches the development server, and you can access your React app at <http://localhost:5173> in your web browser.

- **HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.
- **Version Control:** Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
- Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>

- **Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.
- Visual Studio Code: Download from <https://code.visualstudio.com/download>
 - Sublime Text: Download from <https://www.sublimetext.com/download>
 - WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

Project structure:



The structure of the Trivia Trek application is organized to ensure clarity, modularity, and ease of maintenance. Since the project is developed using **React** along with a **JSON database**, the files and folders are arranged to separate UI components, services, and data-handling layers efficiently. A well-planned directory structure allows developers to collaborate effectively and extend the application with minimal complexity.

src/App.js, App.css, App.jsx (App Component)

These files represent the main App component, which functions as the root of the React application. The App component is responsible for setting up the overall layout, initializing global states, and defining the routing structure. It serves as the entry point from where different pages—such as quiz categories, quiz interface, score summary, and review pages—are rendered.

PROJECT FLOW:-

Project demo:

Before starting to work on this project, let's see the demo.

Demo link: <https://drive.google.com/file/d/1b65uh24M-1Z1aiwgkpdoMs5bhrDPDoIa/view>

Use the code in: <https://github.com/DevLikhith5?tab=repositories>

Milestone 1: Project Setup and Configuration

Installation of Required Tools and Software

To begin development of the Trivia Trek application, the project folder was prepared and all essential tools and libraries were installed. The application is built using React, along with additional packages required for UI design, routing, and interaction with the JSON database.

Tools and Libraries Used:

- React JS
- React Router
- JSON Server (used as a lightweight database)
- Material UI (for UI components)
- React Icons (for UI icons)

These tools collectively enabled the creation of an interactive quiz interface, styled components, and seamless data retrieval from the JSON database.

Reference Resources:

- <https://react.dev/learn/installation>
 - <https://mui.com/material-ui/getting-started/>
 - <https://github.com/typicode/json-server>
-

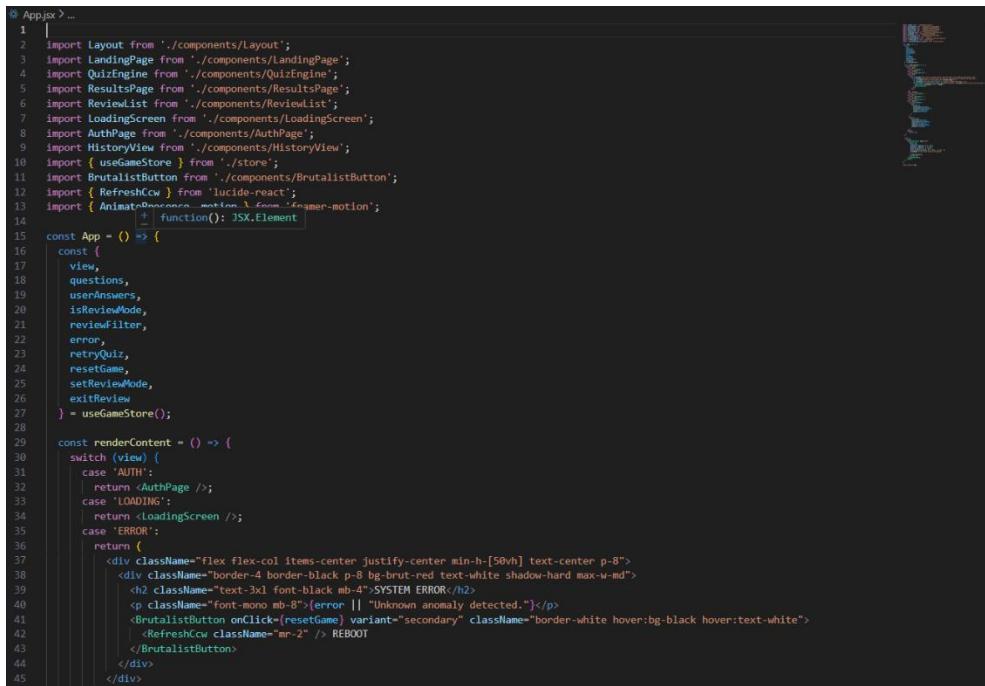
Milestone 2: Web Development

Setup of the React Application

- A new React project was created using the appropriate setup command.
- The file structure was organized to include components, pages, services, and assets.
- Routing was configured using React Router to allow navigation between quiz categories, quiz screens, instructions, and result pages.

- Required libraries such as Material UI, React Icons, and JSON Server were installed to support UI design and backend data management.

App.js component.



```

1  import React from 'react';
2  import Layout from './components/Layout';
3  import LandingPage from './components/LandingPage';
4  import QuizEngine from './components/QuizEngine';
5  import ResultPage from './components/ResultsPage';
6  import ReviewList from './components/ReviewList';
7  import LoadingScreen from './components>LoadingScreen';
8  import AuthPage from './components/AuthPage';
9  import HistoryView from './components/HistoryView';
10 import { useGameStore } from './store';
11 import BrutalistButton from './components/BrutalistButton';
12 import { RefreshCcw } from 'lucide-react';
13 import { AnimatePresence, motion } from 'framer-motion';
14 const App = () => {
15   const {
16     view,
17     questions,
18     userAnswers,
19     isReviewMode,
20     reviewFilter,
21     retryQuiz,
22     error,
23     resetGame,
24     setReviewMode,
25     setRetryMode,
26     exitReview
27   } = useGameStore();
28
29   const renderContent = () => {
30     switch (view) {
31       case 'AUTH':
32         return <AuthPage />;
33       case 'LOADING':
34         return <LoadingScreen />;
35       case 'ERROR':
36         return (
37           <div className="flex flex-col items-center justify-center min-h-[50vh] text-center p-8">
38             <div className="border-4 border-black p-2 bg-brut-red text-white shadow-hard max-w-md">
39               <h2 className="text-3xl font-black mb-4">SYSTEM ERROR</h2>
40               <p className="font-mono mb-8">{error} || "Unknown anomaly detected."</p>
41               <BrutalistButton onClick={resetGame} variant="secondary" className="border-white hover:bg-black hover:text-white">
42                 <RefreshCcw className="mr-2" /> REBOOT
43               </BrutalistButton>
44             </div>
45           </div>
46         );
47     }
48   }

```

Code Description:-

- ./App.css: The CSS file used for styling the App component and controlling the overall layout of the application.
- Imports such as Layout, LandingPage, QuizPage, ResultPage, ReviewList, AuthPage, HistoryView, and BrutalistButton: These components represent different screens and functional sections of the quiz application.
- useGameStore from ./store: A global state management hook used to store and access quiz data, view states, user answers, and other shared information throughout the application.
- The App function is defined, where various states and functions such as view, questions, userAnswers, reviewMode, reviewFilter, retryQuiz, resetGame, and exitReview are retrieved from useGameStore.
- The renderContent() function is used to determine which component should be displayed on screen based on the current view state (for example AUTH, LOADING, ERROR, QUIZ, RESULTS, REVIEW).
- In the ERROR view, a custom error layout is displayed, including an error message and a Reboot button created using the BrutalistButton component.
- The App component returns the Layout component, and inside it the output of renderContent(), ensuring that the appropriate screen is displayed depending on the current state of the application.

- Layout: This component serves as the main structural wrapper for the application, maintaining consistent UI structure across different pages.
 - The App component is exported as the default export of the module, making it available for use in the entry point of the application.
- 2. Design UI components:**

- Create Components.
- Implement layout and styling.
- Add navigation.

3. Implement frontend logic:

- Integration with API endpoints.
- Implement data binding.

Data Provider:-

```
js store.js > ...
1 import { create } from 'zustand';
2
3 import { generateQuizQuestions } from './ai';
4 import { saveQuizResult, fetchUserHistory } from './api';
5
6
7
8
9 export const useGameStore = create((set, get) => ({
10   view: 'AUTH',
11   config: {
12     topic: 'All',
13     difficulty: 'Medium',
14     questionCount: 5,
15     customTopic: ''
16   },
17   score: 0,
18   questions: [],
19   currentQuestionIndex: 0,
20   userAnswers: [],
21   error: undefined,
22   isReviewMode: false,
23   reviewFilter: 'ALL',
24   user: undefined,
25   history: []
26
27   setPendingConfig: (newConfig) => set((state) => ({
28     config: { ...state.config, ...newConfig }
29   })),
30
31   setView: (view) => set({ view }),
32
33   startGame: async () => {
34     const { config } = get();
35
36     if (config.topic === 'Custom' && (!config.customTopic || config.customTopic.trim() === '')) {
37       alert("PLEASE ENTER A CUSTOM TOPIC TO PROCEED.");
38       return;
39     }
40
41     set({ view: 'LOADING', error: undefined });
42
43     try {
44       const questions = await generateQuizQuestions(
45         config
46       );
47
48       set({
49         view: 'QUIZ',
50         config,
51         questions,
52         currentQuestionIndex: 0,
53         userAnswers: [],
54         score: 0
55       });
56     } catch (error) {
57       set({ view: 'ERROR', error });
58     }
59   }
60
61   saveQuizResult: (result) => {
62     const { user } = get();
63
64     if (!user) {
65       return;
66     }
67
68     const updatedUser = {
69       ...user,
70       history: [
71         ...user.history,
72         result
73       ]
74     };
75
76     fetchUserHistory(updatedUser);
77   }
78
79   fetchUserHistory: (user) => {
80     const { config } = get();
81
82     const filteredHistory = user.history.filter((historyItem) => {
83       return config.reviewFilter === 'ALL' || historyItem.topic === config.topic;
84     });
85
86     set({ history: filteredHistory });
87   }
88
89   setScore: (score) => {
90     set({ score });
91   }
92
93   setUserAnswers: (answers) => {
94     set({ userAnswers: answers });
95   }
96
97   setQuestion: (question) => {
98     set({ questions: [question] });
99   }
100
101   setConfig: (config) => {
102     set({ config });
103   }
104
105   setError: (error) => {
106     set({ error });
107   }
108
109   setIsReviewMode: (mode) => {
110     set({ isReviewMode: mode });
111   }
112
113   setReviewFilter: (filter) => {
114     set({ reviewFilter: filter });
115   }
116
117   setHistory: (history) => {
118     set({ history });
119   }
120
121   setCustomTopic: (topic) => {
122     set({ customTopic: topic });
123   }
124
125   setTopic: (topic) => {
126     set({ topic });
127   }
128
129   setDifficulty: (difficulty) => {
130     set({ difficulty });
131   }
132
133   setQuestionCount: (count) => {
134     set({ questionCount: count });
135   }
136
137   setCustomTopic: (topic) => {
138     set({ customTopic: topic });
139   }
140
141   setPendingConfig: (config) => {
142     set({ pendingConfig: config });
143   }
144
145   setView: (view) => {
146     set({ view });
147   }
148
149   setError: (error) => {
150     set({ error });
151   }
152
153   setIsReviewMode: (isReviewMode) => {
154     set({ isReviewMode });
155   }
156
157   setReviewFilter: (reviewFilter) => {
158     set({ reviewFilter });
159   }
160
161   setHistory: (history) => {
162     set({ history });
163   }
164
165   setCustomTopic: (customTopic) => {
166     set({ customTopic });
167   }
168
169   setPendingConfig: (pendingConfig) => {
170     set({ pendingConfig });
171   }
172
173   setView: (view) => {
174     set({ view });
175   }
176
177   setError: (error) => {
178     set({ error });
179   }
180
181   setIsReviewMode: (isReviewMode) => {
182     set({ isReviewMode });
183   }
184
185   setReviewFilter: (reviewFilter) => {
186     set({ reviewFilter });
187   }
188
189   setHistory: (history) => {
190     set({ history });
191   }
192
193   setCustomTopic: (customTopic) => {
194     set({ customTopic });
195   }
196
197   setPendingConfig: (pendingConfig) => {
198     set({ pendingConfig });
199   }
200
201   setView: (view) => {
202     set({ view });
203   }
204
205   setError: (error) => {
206     set({ error });
207   }
208
209   setIsReviewMode: (isReviewMode) => {
210     set({ isReviewMode });
211   }
212
213   setReviewFilter: (reviewFilter) => {
214     set({ reviewFilter });
215   }
216
217   setHistory: (history) => {
218     set({ history });
219   }
220
221   setCustomTopic: (customTopic) => {
222     set({ customTopic });
223   }
224
225   setPendingConfig: (pendingConfig) => {
226     set({ pendingConfig });
227   }
228
229   setView: (view) => {
230     set({ view });
231   }
232
233   setError: (error) => {
234     set({ error });
235   }
236
237   setIsReviewMode: (isReviewMode) => {
238     set({ isReviewMode });
239   }
240
241   setReviewFilter: (reviewFilter) => {
242     set({ reviewFilter });
243   }
244
245   setHistory: (history) => {
246     set({ history });
247   }
248
249   setCustomTopic: (customTopic) => {
250     set({ customTopic });
251   }
252
253   setPendingConfig: (pendingConfig) => {
254     set({ pendingConfig });
255   }
256
257   setView: (view) => {
258     set({ view });
259   }
260
261   setError: (error) => {
262     set({ error });
263   }
264
265   setIsReviewMode: (isReviewMode) => {
266     set({ isReviewMode });
267   }
268
269   setReviewFilter: (reviewFilter) => {
270     set({ reviewFilter });
271   }
272
273   setHistory: (history) => {
274     set({ history });
275   }
276
277   setCustomTopic: (customTopic) => {
278     set({ customTopic });
279   }
280
281   setPendingConfig: (pendingConfig) => {
282     set({ pendingConfig });
283   }
284
285   setView: (view) => {
286     set({ view });
287   }
288
289   setError: (error) => {
290     set({ error });
291   }
292
293   setIsReviewMode: (isReviewMode) => {
294     set({ isReviewMode });
295   }
296
297   setReviewFilter: (reviewFilter) => {
298     set({ reviewFilter });
299   }
300
301   setHistory: (history) => {
302     set({ history });
303   }
304
305   setCustomTopic: (customTopic) => {
306     set({ customTopic });
307   }
308
309   setPendingConfig: (pendingConfig) => {
310     set({ pendingConfig });
311   }
312
313   setView: (view) => {
314     set({ view });
315   }
316
317   setError: (error) => {
318     set({ error });
319   }
320
321   setIsReviewMode: (isReviewMode) => {
322     set({ isReviewMode });
323   }
324
325   setReviewFilter: (reviewFilter) => {
326     set({ reviewFilter });
327   }
328
329   setHistory: (history) => {
330     set({ history });
331   }
332
333   setCustomTopic: (customTopic) => {
334     set({ customTopic });
335   }
336
337   setPendingConfig: (pendingConfig) => {
338     set({ pendingConfig });
339   }
340
341   setView: (view) => {
342     set({ view });
343   }
344
345   setError: (error) => {
346     set({ error });
347   }
348
349   setIsReviewMode: (isReviewMode) => {
350     set({ isReviewMode });
351   }
352
353   setReviewFilter: (reviewFilter) => {
354     set({ reviewFilter });
355   }
356
357   setHistory: (history) => {
358     set({ history });
359   }
360
361   setCustomTopic: (customTopic) => {
362     set({ customTopic });
363   }
364
365   setPendingConfig: (pendingConfig) => {
366     set({ pendingConfig });
367   }
368
369   setView: (view) => {
370     set({ view });
371   }
372
373   setError: (error) => {
374     set({ error });
375   }
376
377   setIsReviewMode: (isReviewMode) => {
378     set({ isReviewMode });
379   }
380
381   setReviewFilter: (reviewFilter) => {
382     set({ reviewFilter });
383   }
384
385   setHistory: (history) => {
386     set({ history });
387   }
388
389   setCustomTopic: (customTopic) => {
390     set({ customTopic });
391   }
392
393   setPendingConfig: (pendingConfig) => {
394     set({ pendingConfig });
395   }
396
397   setView: (view) => {
398     set({ view });
399   }
400
401   setError: (error) => {
402     set({ error });
403   }
404
405   setIsReviewMode: (isReviewMode) => {
406     set({ isReviewMode });
407   }
408
409   setReviewFilter: (reviewFilter) => {
410     set({ reviewFilter });
411   }
412
413   setHistory: (history) => {
414     set({ history });
415   }
416
417   setCustomTopic: (customTopic) => {
418     set({ customTopic });
419   }
420
421   setPendingConfig: (pendingConfig) => {
422     set({ pendingConfig });
423   }
424
425   setView: (view) => {
426     set({ view });
427   }
428
429   setError: (error) => {
430     set({ error });
431   }
432
433   setIsReviewMode: (isReviewMode) => {
434     set({ isReviewMode });
435   }
436
437   setReviewFilter: (reviewFilter) => {
438     set({ reviewFilter });
439   }
440
441   setHistory: (history) => {
442     set({ history });
443   }
444
445   setCustomTopic: (customTopic) => {
446     set({ customTopic });
447   }
448
449   setPendingConfig: (pendingConfig) => {
450     set({ pendingConfig });
451   }
452
453   setView: (view) => {
454     set({ view });
455   }
456
457   setError: (error) => {
458     set({ error });
459   }
460
461   setIsReviewMode: (isReviewMode) => {
462     set({ isReviewMode });
463   }
464
465   setReviewFilter: (reviewFilter) => {
466     set({ reviewFilter });
467   }
468
469   setHistory: (history) => {
470     set({ history });
471   }
472
473   setCustomTopic: (customTopic) => {
474     set({ customTopic });
475   }
476
477   setPendingConfig: (pendingConfig) => {
478     set({ pendingConfig });
479   }
480
481   setView: (view) => {
482     set({ view });
483   }
484
485   setError: (error) => {
486     set({ error });
487   }
488
489   setIsReviewMode: (isReviewMode) => {
490     set({ isReviewMode });
491   }
492
493   setReviewFilter: (reviewFilter) => {
494     set({ reviewFilter });
495   }
496
497   setHistory: (history) => {
498     set({ history });
499   }
500
501   setCustomTopic: (customTopic) => {
502     set({ customTopic });
503   }
504
505   setPendingConfig: (pendingConfig) => {
506     set({ pendingConfig });
507   }
508
509   setView: (view) => {
510     set({ view });
511   }
512
513   setError: (error) => {
514     set({ error });
515   }
516
517   setIsReviewMode: (isReviewMode) => {
518     set({ isReviewMode });
519   }
520
521   setReviewFilter: (reviewFilter) => {
522     set({ reviewFilter });
523   }
524
525   setHistory: (history) => {
526     set({ history });
527   }
528
529   setCustomTopic: (customTopic) => {
530     set({ customTopic });
531   }
532
533   setPendingConfig: (pendingConfig) => {
534     set({ pendingConfig });
535   }
536
537   setView: (view) => {
538     set({ view });
539   }
540
541   setError: (error) => {
542     set({ error });
543   }
544
545   setIsReviewMode: (isReviewMode) => {
546     set({ isReviewMode });
547   }
548
549   setReviewFilter: (reviewFilter) => {
550     set({ reviewFilter });
551   }
552
553   setHistory: (history) => {
554     set({ history });
555   }
556
557   setCustomTopic: (customTopic) => {
558     set({ customTopic });
559   }
560
561   setPendingConfig: (pendingConfig) => {
562     set({ pendingConfig });
563   }
564
565   setView: (view) => {
566     set({ view });
567   }
568
569   setError: (error) => {
570     set({ error });
571   }
572
573   setIsReviewMode: (isReviewMode) => {
574     set({ isReviewMode });
575   }
576
577   setReviewFilter: (reviewFilter) => {
578     set({ reviewFilter });
579   }
580
581   setHistory: (history) => {
582     set({ history });
583   }
584
585   setCustomTopic: (customTopic) => {
586     set({ customTopic });
587   }
588
589   setPendingConfig: (pendingConfig) => {
590     set({ pendingConfig });
591   }
592
593   setView: (view) => {
594     set({ view });
595   }
596
597   setError: (error) => {
598     set({ error });
599   }
600
601   setIsReviewMode: (isReviewMode) => {
602     set({ isReviewMode });
603   }
604
605   setReviewFilter: (reviewFilter) => {
606     set({ reviewFilter });
607   }
608
609   setHistory: (history) => {
610     set({ history });
611   }
612
613   setCustomTopic: (customTopic) => {
614     set({ customTopic });
615   }
616
617   setPendingConfig: (pendingConfig) => {
618     set({ pendingConfig });
619   }
620
621   setView: (view) => {
622     set({ view });
623   }
624
625   setError: (error) => {
626     set({ error });
627   }
628
629   setIsReviewMode: (isReviewMode) => {
630     set({ isReviewMode });
631   }
632
633   setReviewFilter: (reviewFilter) => {
634     set({ reviewFilter });
635   }
636
637   setHistory: (history) => {
638     set({ history });
639   }
640
641   setCustomTopic: (customTopic) => {
642     set({ customTopic });
643   }
644
645   setPendingConfig: (pendingConfig) => {
646     set({ pendingConfig });
647   }
648
649   setView: (view) => {
650     set({ view });
651   }
652
653   setError: (error) => {
654     set({ error });
655   }
656
657   setIsReviewMode: (isReviewMode) => {
658     set({ isReviewMode });
659   }
660
661   setReviewFilter: (reviewFilter) => {
662     set({ reviewFilter });
663   }
664
665   setHistory: (history) => {
666     set({ history });
667   }
668
669   setCustomTopic: (customTopic) => {
670     set({ customTopic });
671   }
672
673   setPendingConfig: (pendingConfig) => {
674     set({ pendingConfig });
675   }
676
677   setView: (view) => {
678     set({ view });
679   }
680
681   setError: (error) => {
682     set({ error });
683   }
684
685   setIsReviewMode: (isReviewMode) => {
686     set({ isReviewMode });
687   }
688
689   setReviewFilter: (reviewFilter) => {
690     set({ reviewFilter });
691   }
692
693   setHistory: (history) => {
694     set({ history });
695   }
696
697   setCustomTopic: (customTopic) => {
698     set({ customTopic });
699   }
700
701   setPendingConfig: (pendingConfig) => {
702     set({ pendingConfig });
703   }
704
705   setView: (view) => {
706     set({ view });
707   }
708
709   setError: (error) => {
710     set({ error });
711   }
712
713   setIsReviewMode: (isReviewMode) => {
714     set({ isReviewMode });
715   }
716
717   setReviewFilter: (reviewFilter) => {
718     set({ reviewFilter });
719   }
720
721   setHistory: (history) => {
722     set({ history });
723   }
724
725   setCustomTopic: (customTopic) => {
726     set({ customTopic });
727   }
728
729   setPendingConfig: (pendingConfig) => {
730     set({ pendingConfig });
731   }
732
733   setView: (view) => {
734     set({ view });
735   }
736
737   setError: (error) => {
738     set({ error });
739   }
740
741   setIsReviewMode: (isReviewMode) => {
742     set({ isReviewMode });
743   }
744
745   setReviewFilter: (reviewFilter) => {
746     set({ reviewFilter });
747   }
748
749   setHistory: (history) => {
750     set({ history });
751   }
752
753   setCustomTopic: (customTopic) => {
754     set({ customTopic });
755   }
756
757   setPendingConfig: (pendingConfig) => {
758     set({ pendingConfig });
759   }
760
761   setView: (view) => {
762     set({ view });
763   }
764
765   setError: (error) => {
766     set({ error });
767   }
768
769   setIsReviewMode: (isReviewMode) => {
770     set({ isReviewMode });
771   }
772
773   setReviewFilter: (reviewFilter) => {
774     set({ reviewFilter });
775   }
776
777   setHistory: (history) => {
778     set({ history });
779   }
780
781   setCustomTopic: (customTopic) => {
782     set({ customTopic });
783   }
784
785   setPendingConfig: (pendingConfig) => {
786     set({ pendingConfig });
787   }
788
789   setView: (view) => {
790     set({ view });
791   }
792
793   setError: (error) => {
794     set({ error });
795   }
796
797   setIsReviewMode: (isReviewMode) => {
798     set({ isReviewMode });
799   }
800
801   setReviewFilter: (reviewFilter) => {
802     set({ reviewFilter });
803   }
804
805   setHistory: (history) => {
806     set({ history });
807   }
808
809   setCustomTopic: (customTopic) => {
810     set({ customTopic });
811   }
812
813   setPendingConfig: (pendingConfig) => {
814     set({ pendingConfig });
815   }
816
817   setView: (view) => {
818     set({ view });
819   }
820
821   setError: (error) => {
822     set({ error });
823   }
824
825   setIsReviewMode: (isReviewMode) => {
826     set({ isReviewMode });
827   }
828
829   setReviewFilter: (reviewFilter) => {
830     set({ reviewFilter });
831   }
832
833   setHistory: (history) => {
834     set({ history });
835   }
836
837   setCustomTopic: (customTopic) => {
838     set({ customTopic });
839   }
840
841   setPendingConfig: (pendingConfig) => {
842     set({ pendingConfig });
843   }
844
845   setView: (view) => {
846     set({ view });
847   }
848
849   setError: (error) => {
850     set({ error });
851   }
852
853   setIsReviewMode: (isReviewMode) => {
854     set({ isReviewMode });
855   }
856
857   setReviewFilter: (reviewFilter) => {
858     set({ reviewFilter });
859   }
860
861   setHistory: (history) => {
862     set({ history });
863   }
864
865   setCustomTopic: (customTopic) => {
866     set({ customTopic });
867   }
868
869   setPendingConfig: (pendingConfig) => {
870     set({ pendingConfig });
871   }
872
873   setView: (view) => {
874     set({ view });
875   }
876
877   setError: (error) => {
878     set({ error });
879   }
880
881   setIsReviewMode: (isReviewMode) => {
882     set({ isReviewMode });
883   }
884
885   setReviewFilter: (reviewFilter) => {
886     set({ reviewFilter });
887   }
888
889   setHistory: (history) => {
890     set({ history });
891   }
892
893   setCustomTopic: (customTopic) => {
894     set({ customTopic });
895   }
896
897   setPendingConfig: (pendingConfig) => {
898     set({ pendingConfig });
899   }
900
901   setView: (view) => {
902     set({ view });
903   }
904
905   setError: (error) => {
906     set({ error });
907   }
908
909   setIsReviewMode: (isReviewMode) => {
910     set({ isReviewMode });
911   }
912
913   setReviewFilter: (reviewFilter) => {
914     set({ reviewFilter });
915   }
916
917   setHistory: (history) => {
918     set({ history });
919   }
920
921   setCustomTopic: (customTopic) => {
922     set({ customTopic });
923   }
924
925   setPendingConfig: (pendingConfig) => {
926     set({ pendingConfig });
927   }
928
929   setView: (view) => {
930     set({ view });
931   }
932
933   setError: (error) => {
934     set({ error });
935   }
936
937   setIsReviewMode: (isReviewMode) => {
938     set({ isReviewMode });
939   }
940
941   setReviewFilter: (reviewFilter) => {
942     set({ reviewFilter });
943   }
944
945   setHistory: (history) => {
946     set({ history });
947   }
948
949   setCustomTopic: (customTopic) => {
950     set({ customTopic });
951   }
952
953   setPendingConfig: (pendingConfig) => {
954     set({ pendingConfig });
955   }
956
957   setView: (view) => {
958     set({ view });
959   }
960
961   setError: (error) => {
962     set({ error });
963   }
964
965   setIsReviewMode: (isReviewMode) => {
966     set({ isReviewMode });
967   }
968
969   setReviewFilter: (reviewFilter) => {
970     set({ reviewFilter });
971   }
972
973   setHistory: (history) => {
974     set({ history });
975   }
976
977   setCustomTopic: (customTopic) => {
978     set({ customTopic });
979   }
980
981   setPendingConfig: (pendingConfig) => {
982     set({ pendingConfig });
983   }
984
985   setView: (view) => {
986     set({ view });
987   }
988
989   setError: (error) => {
990     set({ error });
991   }
992
993   setIsReviewMode: (isReviewMode) => {
994     set({ isReviewMode });
995   }
996
997   setReviewFilter: (reviewFilter) => {
998     set({ reviewFilter });
999   }
1000
1001   setHistory: (history) => {
1002     set({ history });
1003   }
1004
1005   setCustomTopic: (customTopic) => {
1006     set({ customTopic });
1007   }
1008
1009   setPendingConfig: (pendingConfig) => {
1010     set({ pendingConfig });
1011   }
1012
1013   setView: (view) => {
1014     set({ view });
1015   }
1016
1017   setError: (error) => {
1018     set({ error });
1019   }
1020
1021   setIsReviewMode: (isReviewMode) => {
1022     set({ isReviewMode });
1023   }
1024
1025   setReviewFilter: (reviewFilter) => {
1026     set({ reviewFilter });
1027   }
1028
1029   setHistory: (history) => {
1030     set({ history });
1031   }
1032
1033   setCustomTopic: (customTopic) => {
1034     set({ customTopic });
1035   }
1036
1037   setPendingConfig: (pendingConfig) => {
1038     set({ pendingConfig });
1039   }
1040
1041   setView: (view) => {
1042     set({ view });
1043   }
1044
1045   setError: (error) => {
1046     set({ error });
1047   }
1048
1049   setIsReviewMode: (isReviewMode) => {
1050     set({ isReviewMode });
1051   }
1052
1053   setReviewFilter: (reviewFilter) => {
1054     set({ reviewFilter });
1055   }
1056
1057   setHistory: (history) => {
1058     set({ history });
1059   }
1060
1061   setCustomTopic: (customTopic) => {
1062     set({ customTopic });
1063   }
1064
1065   setPendingConfig: (pendingConfig) => {
1066     set({ pendingConfig });
1067   }
1068
1069   setView: (view) => {
1070     set({ view });
1071   }
1072
1073   setError: (error) => {
1074     set({ error });
1075   }
1076
1077   setIsReviewMode: (isReviewMode) => {
1078     set({ isReviewMode });
1079   }
1080
1081   setReviewFilter: (reviewFilter) => {
1082     set({ reviewFilter });
1083   }
1084
1085   setHistory: (history) => {
1086     set({ history });
1087   }
1088
1089   setCustomTopic: (customTopic) => {
1090     set({ customTopic });
1091   }
1092
1093   setPendingConfig: (pendingConfig) => {
1094     set({ pendingConfig });
1095   }
1096
1097   setView: (view) => {
1098     set({ view });
1099   }
1100
1101   setError: (error) => {
1102     set({ error });
1103   }
1104
1105   setIsReviewMode: (isReviewMode) => {
1106     set({ isReviewMode });
1107   }
1108
1109   setReviewFilter: (reviewFilter) => {
1110     set({ reviewFilter });
1111   }
1112
1113   setHistory: (history) => {
1114     set({ history });
1115   }
1116
1117   setCustomTopic: (customTopic) => {
1118     set({ customTopic });
1119   }
1120
1121   setPendingConfig: (pendingConfig) => {
1122     set({ pendingConfig });
1123   }
1124
1125   setView: (view) => {
1126     set({ view });
1127   }
1128
1129   setError: (error) => {
1130     set({ error });
1131   }
1132
1133   setIsReviewMode: (isReviewMode) => {
1134     set({ isReviewMode });
1135   }
1136
1137   setReviewFilter: (reviewFilter) => {
1138     set({ reviewFilter });
1139   }
1140
1141   setHistory: (history) => {
1142     set({ history });
1143   }
1144
1145   setCustomTopic: (customTopic) => {
1146     set({ customTopic });
1147   }
1148
1149   setPendingConfig: (pendingConfig) => {
1150     set({ pendingConfig });
1151   }
1152
1153   setView: (view) => {
1154     set({ view });
1155   }
1156
1157   setError: (error) => {
1158     set({ error });
1159   }
1160
1161   setIsReviewMode: (isReviewMode) => {
1162     set({ isReviewMode });
1163   }
1164
1165   setReviewFilter: (reviewFilter) => {
1166     set({ reviewFilter });
1167   }
1168
1169   setHistory: (history) => {
1170     set({ history });
1171   }
1172
1173   setCustomTopic: (customTopic) => {
1174     set({ customTopic });
1175   }
1176
1177   setPendingConfig: (pendingConfig) => {
1178     set({ pendingConfig });
1179   }
1180
1181   setView: (view) => {
1182     set({ view });
1183   }
1184
1185   setError: (error) => {
1186     set({ error });
1187   }
1188
1189   setIsReviewMode: (isReviewMode) => {
1190     set({ isReviewMode });
1191   }
1192
1193   setReviewFilter: (reviewFilter) => {
1194     set({ reviewFilter });
1195   }
1196
1197   setHistory: (history) => {
1198     set({ history });
1199   }
1200
1201   setCustomTopic: (customTopic) => {
1202     set({ customTopic });
1203   }
1204
1205   setPendingConfig: (pendingConfig) => {
1206     set({ pendingConfig });
1207   }
1208
1209   setView: (view) => {
1210     set({ view });
1211   }
1212
1213   setError: (error) => {
1214     set({ error });
1215   }
1216
1217   setIsReviewMode: (isReviewMode) => {
1218     set({ isReviewMode });
1219   }
1220
1221   setReviewFilter: (reviewFilter) => {
1222     set({ reviewFilter });
1223   }
1224
1225   setHistory: (history) => {
1226     set({ history });
1227   }
1228
1229   setCustomTopic: (customTopic) => {
1230     set({ customTopic });
1231   }
1232
1233   setPendingConfig: (pendingConfig) => {
1234     set({ pendingConfig });
1235   }
1236
1237   setView: (view) => {
1238     set({ view });
1239   }
1240
1241   setError: (error) => {
1242     set({ error });
1243   }
1244
1245   setIsReviewMode: (isReviewMode) => {
1246     set({ isReviewMode });
1247   }
1248
1249   setReviewFilter: (reviewFilter) => {
1250     set({ reviewFilter });
1251   }
1252
1253   setHistory: (history) => {
1254     set({ history });
1255   }
1256
1257   setCustomTopic: (customTopic) => {
1258     set({ customTopic });
1259   }
1260
1261   setPendingConfig: (pendingConfig) => {
1262     set({ pendingConfig });
1263   }
1264
1265   setView: (view) => {
1266     set({ view });
1267   }
1268
1269   setError: (error) => {
1270     set({ error });
1271   }
1272
1273   setIsReviewMode: (isReviewMode) => {
1274     set({ isReviewMode });
1275   }
1276
1277   setReviewFilter: (reviewFilter) => {
1278     set({ reviewFilter });
1279   }
1280
1281   setHistory: (history) => {
1282     set({ history });
1283   }
1284
1285   setCustomTopic: (customTopic) => {
1286     set({ customTopic });
1287   }
1288
1289   setPendingConfig: (pendingConfig) => {
1290     set({ pendingConfig });
1291   }
1292
1293   setView: (view) => {
1294     set({ view });
1295   }
1296
1297   setError: (error) => {
1298     set({ error });
1299   }
1300
1301   setIsReviewMode: (isReviewMode) => {
1302     set({ isReviewMode });
1303   }
1304
1305   setReviewFilter: (reviewFilter) => {
1306     set({ reviewFilter });
1307   }
1308
1309   setHistory: (history) => {
1310     set({ history });
1311   }
1312
1313   setCustomTopic: (customTopic) => {
1314     set({ customTopic });
1315   }
131
```

generation and saving user data.

- Creates the useGameStore state manager using Zustand, which holds all key variables needed for the quiz system, such as:
 - view: Controls which screen (Auth, Loading, Quiz, Result, Review) is displayed.
 - config: Stores quiz settings including topic, difficulty, and question count.
 - score, questions, currentQuestionIndex, userAnswers: Handle the active quiz session.
 - error, isReviewMode, reviewFilter, user, history: Additional states used for UI flow and user data.
- Provides functions such as setPendingConfig to update quiz settings, and setView to change the current screen.
- Defines startGame as an asynchronous function that validates custom topic input, switches the view to Loading, generates quiz questions using generateQuizQuestions, and initializes the quiz state.

DB Component:-

```
1  {
2    "results": [
3      {
4        "id": "90b9",
5        "userId": "google-1763790979714",
6        "topic": "about iron",
7        "difficulty": "Easy",
8        "score": 5,
9        "totalQuestions": 5,
10       "timestamp": "2025-11-22T05:57:23.668Z"
11      },
12      {
13        "id": "12b4",
14        "userId": "google-1763792504617",
15        "topic": "Generate 10 Java SE 17 OCP-style multiple-choice questions covering modules, \nrecords, sealed classes, lambdas, streams, gener
16        "difficulty": "Medium",
17        "score": 2,
18        "totalQuestions": 5,
19        "timestamp": "2025-11-22T06:23:17.098Z"
20      },
21      {
22        "id": "30e5",
23        "userId": "google-1763792745306",
24        "topic": "questions on golang",
25        "difficulty": "Medium",
26        "score": 5,
27        "totalQuestions": 5,
28        "timestamp": "2025-11-22T06:27:01.622Z"
29      },
30      {
31        "id": "2921",
32        "userId": "google-1763793524502",
33        "topic": "Geography",
34        "difficulty": "Medium",
35        "score": 11,
36        "totalQuestions": 20,
37        "timestamp": "2025-11-22T06:40:53.224Z"
38      },
39      {
40        "id": "3a36",
41        "userId": "google-1763793524502",
42        "topic": "quantum physics",
43        "difficulty": "Easy",
44        "score": 5,
45        "totalQuestions": 5,
```

Code Description:-

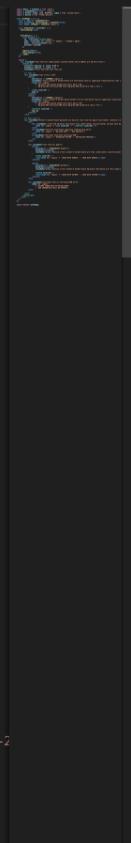
- The db.json file serves as the local JSON database for the application, used through JSON Server to simulate a backend.
- It contains an array named "results", which stores all quiz attempts made by users.
- Each object inside the "results" array represents a single quiz result entry.
- Every entry includes the following fields:
 - id: A unique identifier automatically assigned to each result.
 - userId: Identifies the user who attempted the quiz.
 - topic: The topic used for generating the quiz.
 - difficulty: The difficulty level selected by the user.
 - score: The number of correct answers obtained.
 - totalQuestions: Total number of questions in the quiz.
 - timestamp: Stores the date and time when the quiz was completed.
- This structure allows the application to store past quiz history and retrieve it when needed, such as for showing the user's previous results.
- JSON Server uses this file to provide REST API endpoints (GET, POST, etc.) for interacting with quiz results during development.

Auth Page Component:-

```

1 import React, { useState } from 'react';
2 import { useGameStore } from '../store';
3 import { Github, Globe, Lock, UserPlus, LogIn } from 'lucide-react';
4 import { motion } from 'framer-motion';
5
6 const AuthPage = () => {
7   const { login } = useGameStore();
8   const [isloading, setIsLoading] = useState(false);
9   const [mode, setMode] = useState('signin');
10
11   const handleAuth = (provider) => {
12     setIsLoading(true);
13
14
15     setTimeout(() => {
16       const mockUser = {
17         id: `${provider}-${Date.now()}`,
18         name: `${provider === 'google' ? 'Google' : 'GitHub'} Agent`,
19         email: `agent@${provider}.com`,
20         provider: provider
21       };
22
23
24       login(mockUser);
25       setIsLoading(false);
26     }, 1000);
27   };
28
29
30   return (
31     <div className="flex flex-col items-center justify-center min-h-[80vh] p-4 md:p-8 w-full">
32       <motion.div
33         initial={{ opacity: 0, scale: 0.95 }}
34         animate={{ opacity: 1, scale: 1 }}
35         className="w-full max-w-md flex flex-col"
36       >
37         {/* Tabs */}
38         <div className="flex w-full z-10">
39           <button
40             onClick={() => setMode('signin')}
41             className={flex-1 border-4 border-black p-4 font-black uppercase transition-all flex items-center justify-center gap-2
42               ${mode === 'signin'
43                 ? 'bg-white text-black border-b-0 top-1 pb-5 z-20'
44                 : 'bg-gray-200 text-gray-500 hover:bg-gray-300 border-b-4 top-1 z-0'}}>
45             >
46               <LogIn size={20} />
47             </button>
48           <button
49             onClick={() => setMode('signout')}
50             className={flex-1 border-4 border-black p-4 font-black uppercase transition-all flex items-center justify-center gap-2
51               ${mode === 'signout'
52                 ? 'bg-white text-black border-b-0 top-1 pb-5 z-20'
53                 : 'bg-gray-200 text-gray-500 hover:bg-gray-300 border-b-4 top-1 z-0'}}>
54             >
55               <SignOut size={20} />
56             </button>
57           </div>
58         </div>
59       </motion.div>
60     </div>
61   );
62 }

```



Code Description:-

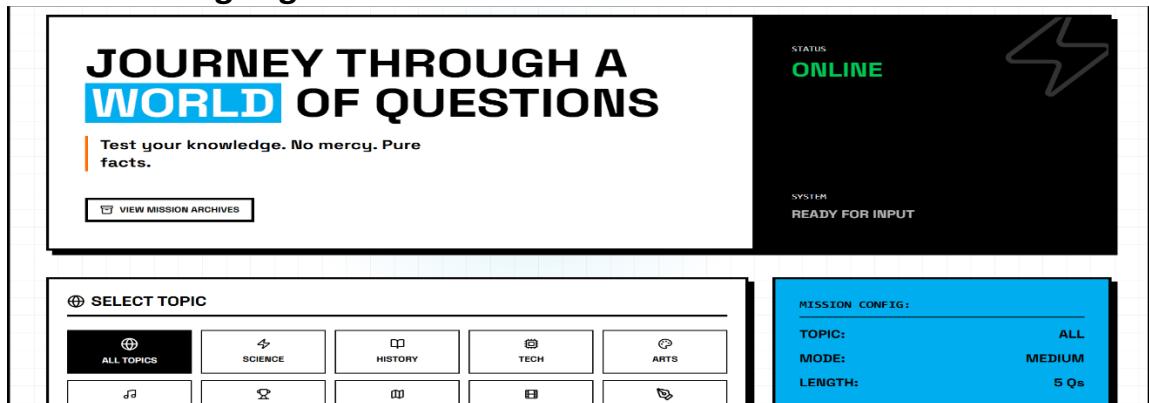
- Imports React, useState, and motion for UI animations, along with useGameStore for accessing global state and login functionality.
- Imports icon components such as Github, Globe, Lock, UserPlus, and LogIn from lucide-react to enhance the visual layout of the authentication screen.
- Defines the AuthPage functional component, which manages the login interface for the application.
- Retrieves the login function from useGameStore, which is used to update the global user state after authentication.
- Uses two local state variables, isLoading and mode.
 - isLoading controls the animation and delay during the login simulation.
 - mode switches between different authentication modes such as “signin.”
- Defines the handleAuth function, which accepts a provider name (such as Google or GitHub) and simulates a login process by enabling loading and creating a mock user object after a timeout.
- The mock user object contains id, name, email, and provider information based on the selected provider.
- After creating the mock user, the login function updates the global store, and loading is disabled.

Project Execution:

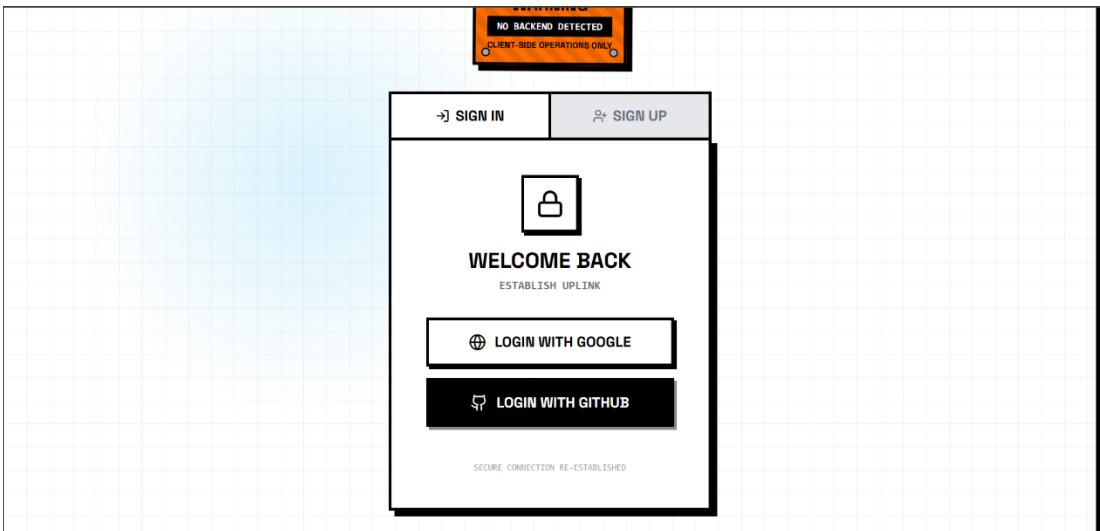
After completing the code, run the react application by using the command “npm start” or “npm run dev” if you are using vite.js

Here are some of the screenshots of the application.

- **Landing Page**



- **Auth Page Component**



- **Select Topic Component**

The image displays four components arranged vertically:

- SELECT TOPIC:** A grid of ten categories: ALL TOPICS, SCIENCE, HISTORY, TECH, ARTS, MUSIC, SPORTS, GEOGRAPHY, MOVIES, and CUSTOM. The "ALL TOPICS" button is highlighted with a black background.
- DIFFICULTY:** A dropdown menu with options: EASY, MEDIUM (highlighted with an orange background), HARD, and BRUTAL.
- QUANTITY:** A grid of four numbers: 5, 10, 15, and 20. The "5" button is highlighted with a black background.
- MISSION CONFIG:** A section with three settings: TOPIC (ALL), MODE (MEDIUM), and LENGTH (5 Qs).

At the bottom right is a large black button labeled "INITIATE ▶".

- **Questions Component**

- **Results Component**

Project Demo link:

<https://drive.google.com/file/d/1HmliPpQAQOax-VxmouLfrGV1crcH1ZaH/view>