Quiz Game

SaiLahari (016037112), Sai Manasa Yadlapalli (015999659), Mounica Reddy Kandi (016021902), Chandu V (016085251)

Department of Computer Engineering
San Jose State University
San Jose, CA, USA

sailahari.seethamraju@sjsu.edu, saimanasa.yadlapalli@sjsu.edu, mounicareddy.kandi@sjsu.edu, satyavenkatachandratej.ventrapragada@sjsu.edu

GitHub: https://github.com/sjsucmpe272SP22/Quiz-Game

***Abstract* — This project is a Full Stack implementation of a real-time single-player Quiz Game. Data fetch is from an Open-source context API 'Trivia'. The Player has the option to choose the number of questions before starting a game. Questions are multiple-choice and the highest score gets displayed on the Leader board.**

Keywords—Data fetch, API, Full-stack, Trivia

I. INTRODUCTION

Quiz Game is a simple and fun game designed for Entertainment purposes where a user is provided an interactive platform to take quizzes on various categories. The game is based on the idea of representing data from the third-party Open Trivia API in the form of questions and answers for the game. The application is divided into Homepage with Top10 players, Login and Register pages with authentication, Game page, Account page to display Player statistics and game settings and an About page to display README.md file of our GitHub repository.

The following is our analysis of system architecture.

II. ARCHITECTURE

Our application architecture comprises the following major components: NextJS, NodeJS, and Mongo DB for the database. NextJS is a popular React framework on GitHub, which is used for static websites, Desktop, Lightweight Apps, Pre-rendered apps Mobile Web, etc. In the production stage, it optimizes the end users experience. There are many advantages of using Next.js, such as easier Pre-Rendering, exporting a static website with a single command, including CSS in JS code, very less configuration, fully extensible, and optimized for smaller builds, etc. We used Nodejs for event-driven server communication. Also, we specifically decided to use MongoDB for its ease of Schema collection use and Mongo Atlas to host and manage user-provided data in the cloud.

The data request flow begins when a player creates a new account in our web application. Registration and sign-in have been merged into one page for easy access. After creating an account, the Player can choose any avatar provided (we used female and male avatars). The player can also choose the number of questions in the game even before starting a new game. We decided to give 3 questions as a minimum and 20 as a maximum to choose from. Three difficulty level questions will pop up in the quiz and the score changes for each level.

The leaderboard consists of the top 10 players' data displayed. This is being performed by Server-Side Rendering feature in NextJS. The leader board page can be viewed without even logging into the application. Values in the top 10 positions of the leaderboard change as players play new games and the score changes responsively. All the data related to the player such as name, password, statistics, and preferences will be saved on the mongo atlas cloud from Node. Used JWT tokens for password authentication and bcrypt for password hashing. This improves player account privacy.

The Application is hosted on Vercel.com for easy convincing. We chose Vercel over other cloud services to deploy because Vercel is created & owned by the same team that is behind NextJS.

Below is our full stack application architectural flow representation.

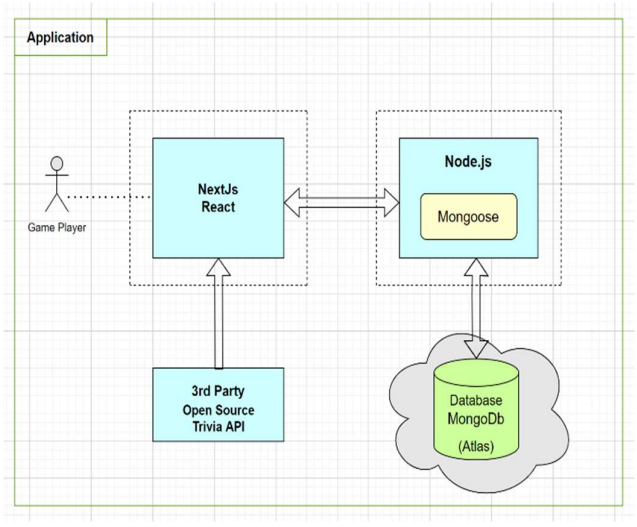


Fig. 1. Application Architecture diagram

Below is our Use Case UML diagram. Here, the Game player is our Actor, displaying high-level functional interactions & requirements of the web application from a player perspective.

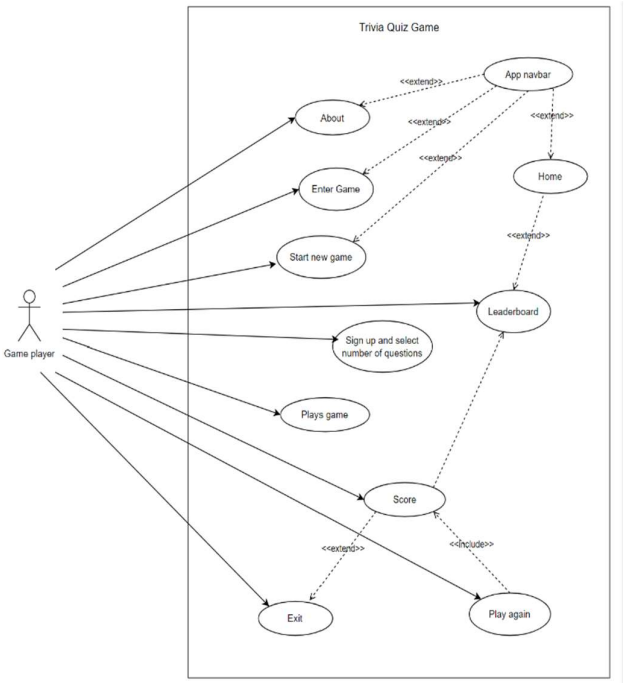


Fig. 2. UML Use Case diagram

III. TRIVIA OPEN API

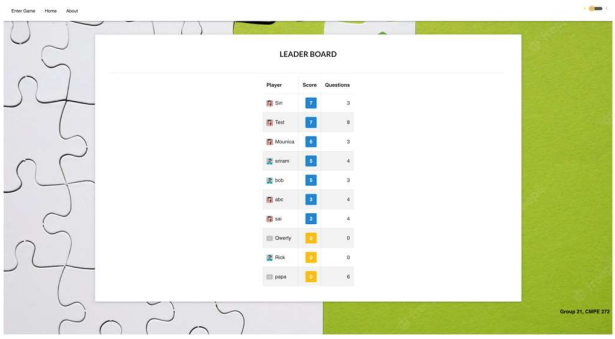
The data is fetched from Trivia API in real-time for quiz questions and answers. We chose to implement Trivia API (opentdb.com) as it is free to use and has a user-contributed question database. The questions are from various categories like Science, Computers, Entertainment (Video Games), General Knowledge, Mathematics, Geography, Comics, Television, Cartoon and Animation, Music, Books, Films, Sports, Japanese Anime and Manga, History, Vehicles, Nature, Mythology, Celebrities, Politics, Gadgets, Art, Animals, Musicals and Theatres.

IV. Quiz Game Web Application

To create a game genre web application which doesn't have any kind of limitations to every age group of users and all subjects, we decided to develop a QUIZ GAME. When the user's login to the application, they can choose the difficulty level of questions based on their requirement. User can select their avatar either "male" or "female" while setting up the account. All the available User Interface pages of web application are listed below.

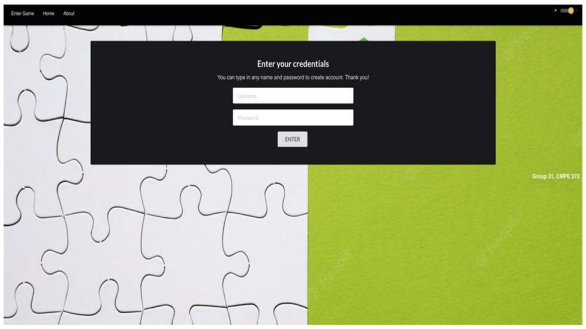
1) Quiz Game Homepage

Leaderboard can be viewed on the homepage with player details and the scores of the top 10 players without signing into the application. When a user requests a webpage, the server prepares the page by fetching user-specific data and sending it to the user's machine. The browser then construes the content and displays the page. This works because of the server-side rendering functionality of NextJS. There are two themes currently available to view our application. They are Dark and Light themes. Each user can change the theme of the application at any time during the game.



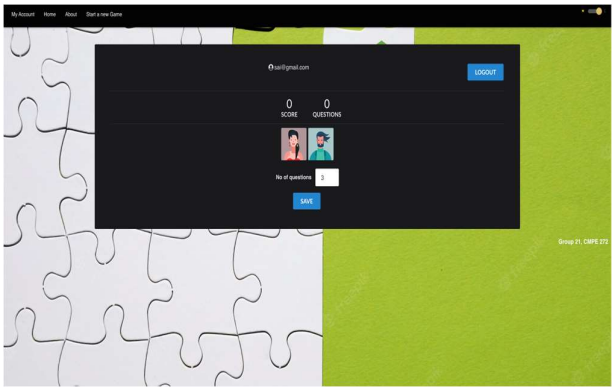
2) Game Login

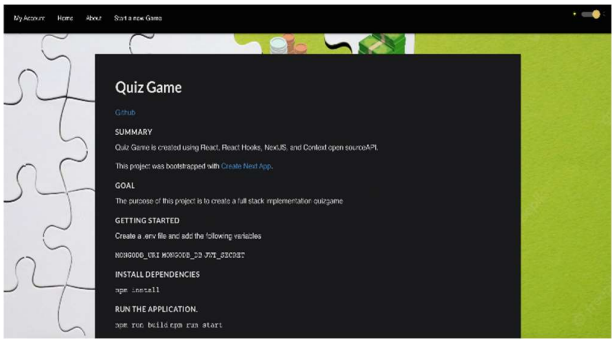
When the user selects "Enter Game" on the navbar, a login page gets displayed. If he/she is new to the game, an account should be created in order to play. All existing users can use their credentials to login and play. When the user login to the application, he/she will be directly navigated to the homepage.



3) User Profile

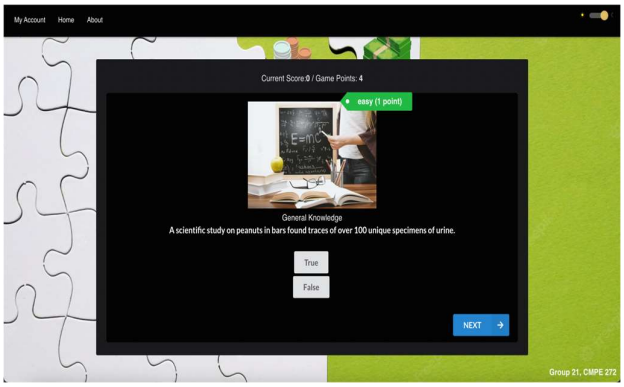
Whenever the user login into the quiz game, he/she can choose "MyAccount" option which is on the top left side of the navbar. Each user can select the total number of questions that they want to play in a game where he/she should play at least 3 questions and at most 20 questions. The total score of the game depends on the number of questions that user chooses to answer and their difficulty levels. At this point of time, user has two options. After the account setup, if user wish to play the game, he/she can proceed and select "Start a new Game" on the navbar. The other option is from the user profile page, he/she can directly logout of the quiz game web application and comeback later to play the game.





4) Start a New Game

Clicking on the "start a new game" enters you into the game with the number of questions that you have opted to play. The questions you obtain are divided into categories like general knowledge, entertainment, history, geography, and many more. These questions come in randomly based on the categories listed and the complexity also varies i.e., easy, medium, and hard which have points 1, 2, and 3 respectively for each question. On top, you can view the current score and the game points. You will have a question displayed with the options below.



5) About Page

The About Page displays all the information related to the application. This page is generated at build time and will be reused on each request. We are displaying static data by using NextJS static generation.

V. LINKS

[1] GitHub: https://github.com/sjsucmpe272SP22/Quiz-Game

[2] Hosted application: https://quiz-game-cmpe-272.vercel.app/

VI. REFERENCES

[1] https://nextjs.org/

[2] https://vercel.com/

[3] https://reactjs.org/

[4] https://nodejs.org/en/

[5] https://www.mongodb.com/atlas/database

[6] https://opentdb.com/api_config.php

[7] https://react.semantic-ui.com/

[8] https://www.npmjs.com/package/bcrypt