Khulna University of Engineering and Technology

Project Report on QuizWiz: A Quiz and Trivia App

Course Name: Mobile Computing Laboratory **Course Code: CSE 3218**

Submitted From	**Submitted To**
Naimur Rahman Roll: 1907031	Most. Kaniz Fatema Isha Lecturer
Md. Atique Faisal Roll: 1907038	Department of Computer Science and Engineering, KUET
Zobayer Abedin Roll: 1907045	Argha Chandra Dhar Lecturer
Asfaq Mahmud Saif Roll: 1907046	Department of Computer Science and Engineering, KUET
Md. Zakaria Hossain Roll: 1907056	Date of Submission: 28-11-2023

1.1 - Introduction:

In the dynamic landscape of mobile applications, QuizWiz, the Quiz/Trivia App developed using SwiftUI stands as an engaging and interactive platform designed to challenge and entertain users. This project seamlessly integrates SwiftUI to create an immersive and visually appealing user interface, while incorporating a range of features to enhance the overall user experience.

Key Features:

- ✓ **User Sign In/Sign Up:** Users can sign in securely, ensuring a personalized experience. New users have the option to sign up, providing a gateway to exclusive features.
- ✓ **Custom Quiz Creation:** Empowering users with creativity, the app allows them to craft their quizzes with custom questions and answers. A user-friendly interface facilitates easy quiz creation, generating a unique Quiz ID to keep track of the created quiz.
- ✓ **Custom Quiz Joining:** Users can seamlessly join custom quizzes created by others using the Quiz ID, promoting a sense of community and competition. The app utilizes intuitive navigation to facilitate easy exploration and participation in diverse quizzes.
- ✓ **API-Based Trivia:** Leveraging external API, named Open Trivia Database, the app diversifies its question database to ensure a wide array of topics and challenges. The integration of APIs enhances the app's content richness, keeping users engaged with fresh and interesting trivia. Json parsing is used to take data from the API to the UI.
- ✓ **Recent Result Tracking:** The app maintains the record of user performance, including scores and Quiz ID for the most recent user created participated quiz. Results are stored securely, allowing users to track their recent progress.
- ✓ **Target Audience:** This Quiz/Trivia App caters to a diverse audience, ranging from avid quiz enthusiasts to casual gamers seeking a fun and intellectually stimulating experience. The app's versatility in custom quiz creation and joining, coupled with API-driven trivia content, ensures broad appeal across different demographics.
- ✓ **User Interface:** The project harnesses the power of SwiftUI, Apple's modern and declarative UI framework, to create a seamless and visually compelling user interface.

Key Components:

- ✓ SwiftUI
- ✓ Firebase Database
- ✓ Json Parsing
- ✓ Open Trivia Database API

In the subsequent sections, we will delve into the technical intricacies of the app's implementation, challenges, and the UI employed to make QuizWiz: A Quiz & Trivia App a standout contribution to the realm of mobile applications.

1.2 - Objectives and Goals:

1. Develop a user-friendly quiz/trivia app interface using SwiftUI
2. Implement secure sign-in/sign-up functionality for user authentication
3. Enable users to create and join custom quizzes for a personalized experience
4. Integrate external API to diversify trivia content and enhance question variety
5. Establish a comprehensive system for storing and displaying most recent user quiz results
6. Optimize app performance and responsiveness for a seamless user experience
7. Use Firebase to ensure proper data storage and data fetching
8. Use Jason Parsing to seamlessly extract API Jason based data into the app
9. Develop an UI that enables the user to get and set exactly how much is needed to fulfil the user experience
10. Set proper indicators that enables users to seamlessly set and view the correct and incorrect answers

2. Methodology:

The entire project can be divided into 6 separate parts:

1. Sign In/Sign Up
2. Homepage
3. Create Quiz
4. Join Quiz
5. Trivia
6. Results

2.1 - Sign In/Sign Up:

2.1.1 - Sign Up:

This portion can be divided into 2 separate parts:

1. User Interface
2. Firebase Data Storing

2.1.1.1 - User Interface:

Consists of 3 separate text fields:

1. **Username:** contains the new username entered by the user.
2. **Email:** contains the email provided by the user.

3. **Password:** contains the preferred password for the user.

Also 2 buttons:

1. **Sign Up:** stores the data in the text fields into the firebase and thus creating a new user for the login system.
2. **Sign In:** redirects to the sign in page.

2.1.1.2 - Firebase Data Storing:

Once clicked on the sign-up button the data in the text fields are stored into the firebase authentication system. Here all the text fields are automatically converted to firebase authentication preferred data. The email and password are stored into the authentication system whereas the username is stored into the firebase real time database instead. This was done to handle future methods regarding the username but no such methods were implemented later.

2.1.2 - Sign In:

This portion can be divided into 2 separate parts:

1. User Interface
2. Firebase Data Fetching

2.1.2.1 - User Interface:

Consists of 2 separate text fields:

1. **Email:** contains the email by the user that is related to the user's authentication which was entered while signing up.
2. **Password:** contains the password used by the user while signing up.

Also 2 buttons:

1. **Sign In:** gets the data in the text fields and compares with the firebase stored data trying to find the user by email and also match the password.
2. **Sign Up**: redirects to the sign-up page.

2.1.2.2 - Firebase Data Fetching:

Once clicked on the sign in button the data in the text fields are compared with the data stored in the firebase authentication system. The email is used as a key to search the authentication fields and upon the match being found with the key, the password is checked with the hashed password already stored in the authentication system. If the key in email is not found in the database, then it returns an error and same if the password is not matched with the stored one.

2.2 - Homepage:

The homepage consists of 4 sections:

1. The Create Quiz Button: redirects to the create quiz section
2. The Join Quiz Button: redirects to the join quiz section
3. The Trivia Button: redirects to the trivia section
4. The Most Recent User Created Quiz Results: shows the result for that user's most recent user created quiz participation score.

2.3 - Create Quiz:

In this page, the user will be able to create self-made quiz that can be later be participated in by other users. This part can be divided into 6 parts:

1. Quiz Id
2. Question Text Field
3. Answers Text Fields
4. Correct Answer Choice
5. Add Button
6. Back Button

2.3.1 - Quiz Id:

This is automatically generated id by the system based on timestamps. It is represented as the quiz id for later and this key will be used to join into the quiz. This id must be kept by the creator of the quiz in order for people to join in the quiz. The quiz id is shown at the bottom of the screen and remains constant throughout this session of adding quiz.

2.3.2 - Question Text Field:

In this section, the user will be able to add the desired question that he wants to add into the questions. This is stored as a string variable to the database under the quiz id generated previously.

2.3.3 - Answers Text Fields:

Here, the user will have 4 options to add the answers to the question he provided. All 4 fields must be unique and be filled. Otherwise, the app might cause an error. These are stored as keys for 4 sets of maps in to the database under the question.

2.3.4 - Correct Answer Choice:

A checkmark is given beside each answer text fields to denote the correct answer. Here, the all the answers will be represented with a red cross at the start for all fields. To denote the correct answer the user needs to select the checkmark and upon selection the checkmark will change into a green tik mark to represent the correct answer. The user must choose one correct answer in order to avoid future errors. These are represented as bool values to the keys of answers into the database under the questions.

2.3.5 - Add Button:

This button adds the question and all the answers to the firebase database under the "quizes" node. First it will take into account the quiz id and then put the question under that as a node and put 4 maps of answers and choices under it as leaf nodes. Thus, the question is stored into the database and it will clear all the text fields and checkmarks and continue to add more questions under the same quiz id until back button is initiated.

2.3.6 - Back Button:

This is essentially the finish button for adding quiz questions. When clicked it will return back to home and thus end the sequence of questions as the complete quiz under that specific quiz id.

2.4 - Join Quiz:

This section can be divided firstly into 2 separate stages:

1. Id Entry Stage
2. Quiz Stage

2.4.1 - Id Entry Stage:

In this stage there are 2 components being:

1. Id Text Field
2. Join Button

The id of the quiz that the user wants to join is firstly written into the text field and then the join button is clicked. If there is an id corresponding to the provided id for quiz, then the user is redirected into that quiz. Otherwise, it is shown that the id is not found.

2.4.2 - Quiz Stage:

In this stage there are 5 different working components:

1. Question and Answer Options
2. Selection of Answer
3. Next Button
4. Progress Bar and Progress Counter
5. Correct Answer Count

2.4.2.1 - Question and Answer Options:

Here the question and the 4 different answers are shown. They are fetched from the firebase real time database based on the quiz id provided. They are fetched in a random order from the database. So, the sequence generated will be different each time for a joined quiz and the same goes for the answer options as well of being shuffled in a random order.

2.4.2.2 - Selection of Answer:

The user can select an option from the 4 options provided based upon their guess or knowledge about the correct answer. The selected answer is compared with the correct answer provided in the database and if the answer is correct then a green check mark is shown. On the other hand, if the answer is wrong then a red cross mark is shown to denote that the user has given the wrong answer. Each time the user answers the selected answer is highlighted and the unselected answers grey out denoting the selection of the user.

2.4.2.3 - Next Button:

The next button simply shows the next question of the quiz and upon ending the quiz shows the result page. The button initially can't be selected if there is no answer selected by the user and only be selected when the user chooses to answer a question.

2.4.2.4 - Progress Bar and Progress Counter:

This section shows the progress of the quiz in 2 different formats. The bar shows it in a graphical manner whereas the counter shows it in numeric manner with the length of the quiz and as well how many questions have been answered at this point.

2.4.2.5 - Correct Answer Count:

The number of correct answers given by the user is kept in a variable and in the end is shown in the result section.

2.5 - Trivia:

This is the section where an API is used to create a basic trivia game. The API that is used here is called "Open Trivia Database". This API can generate random trivia based on various categories. The categories can be of many types and based on the selected categories, the API generates a Json based output that can be parsed into applications.

The configuration that is used in the app is given as such:

- ✓ Number of Questions: 10
- ✓ Category: Any
- ✓ Difficulty: Any

- ✓ Type: Any
- ✓ Encoding: Default Encoding

Based on this configuration a model is generated by the API and that model is also implemented into the app. The structure of this model goes as follows:

- ✓ Type
- ✓ Difficulty
- ✓ Category
- ✓ Question
- ✓ Correct Answer
- ✓ Incorrect Answers

This model is parsed into the app using Json parsing and using the model is set into the UI for showing.

The trivia section can be divided into 2 different sections:

1. The Question/Answer Section
2. The Result Section

2.5.1 - The Question/Answer Section:

This is similar to the join quiz's quiz stage with basically the same UI. The key difference is that here there are not set answer options. Instead, they come from the API as variable options. So, there could be from 2 to 4 different options to chose from based on the question and answers generated by the API.

Here, no database is involved in any shape or form. Instead, everything is done is almost an Model View Control or MVC manner.

Model:

As previously discussed, it creates a model in Json form and thus a similar model needs to be created in the app to take those data and parse into the model.

View:

The view section consists of the UI and is exactly the same as join quiz.

Control:

This section can be the logical section. The trivia is managed by a trivia manager that handles all the data. It fetches the data from the API and sets it to the model. Then it passed the model into the view. Also, it handles all the corner cases and all the various errors.

2.5.2 - The Result Section:

In this section, the correct number of answers given by the user is shown. It is kept track of by a variable in the manager. This also includes 2 buttons:

1. The Play Again Button: immediately starts another trivia game
2. The Back Button: backs into the homepage

Here, the result is not stored into the database. So, the result is a one time thing and is temporary. It is only used as a game rather than being a full result tracking quiz.

2.6 - Results:

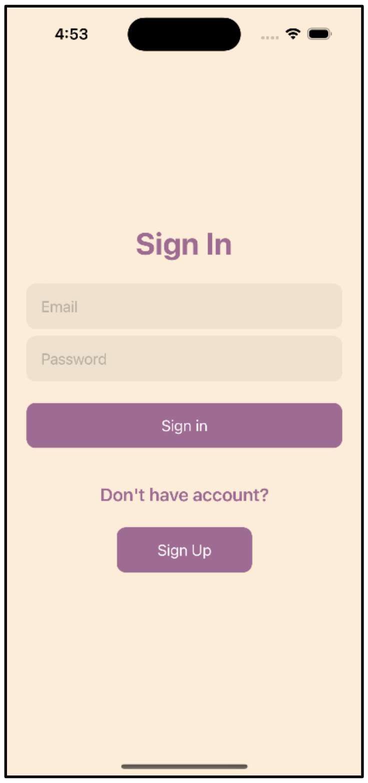
This field mainly works for a user created quiz and not an API based trivia. The result is shown to screen after the completion of a user created quiz. Also, the score along with the quiz id is pushed into the database under that user. In this case, only the most recent score and quiz id is stored into the database. After backing out, this score is also shown in the home section with the quiz id and score.

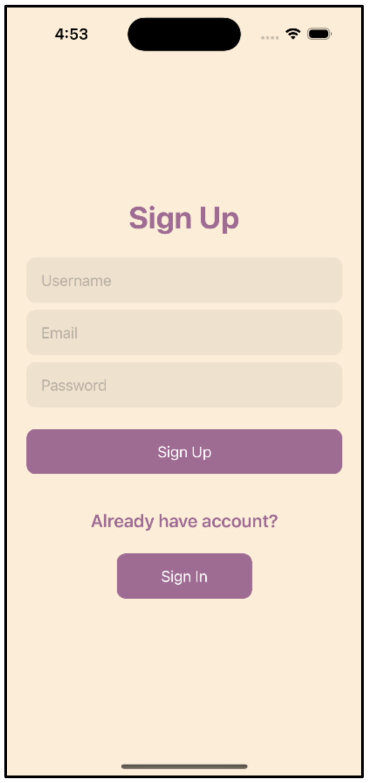
The score is stored in a 3-node format as follows:

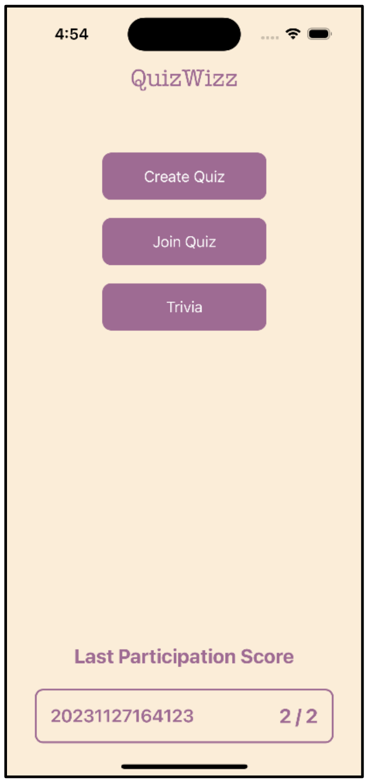
- ✓ Quiz ID
- ✓ Score
- ✓ Length

This is stored under userinfo -> auth id -> participated as a key value map with 3 leaf nodes. This data is fetched from the database to the homepage upon appearance of the page.

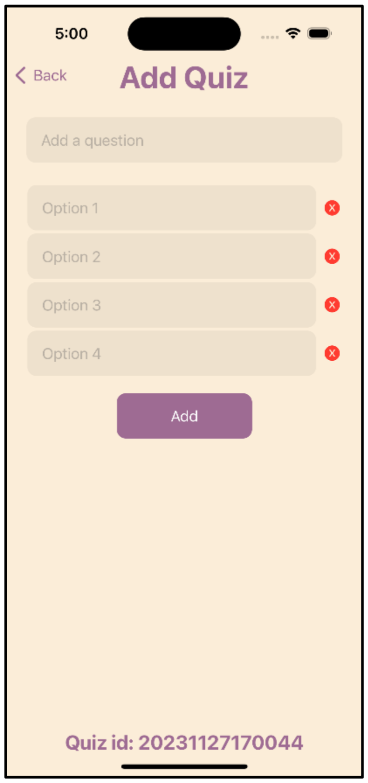
3. Results and Screenshots:



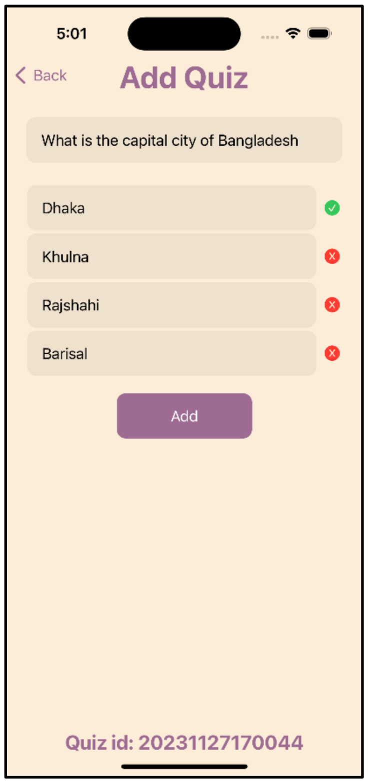




20231127164123 2 / 2



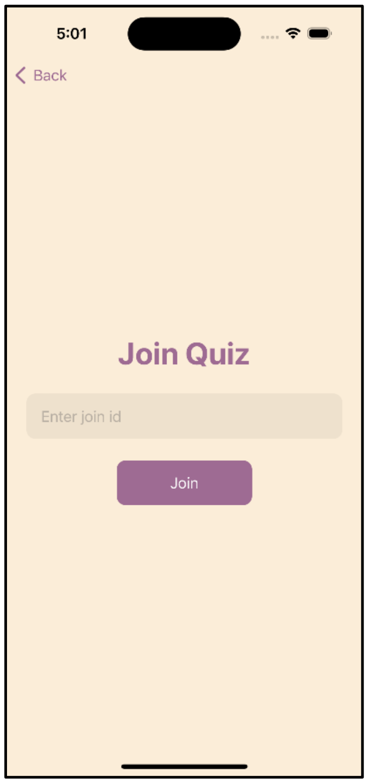
Quiz id: 20231127170044

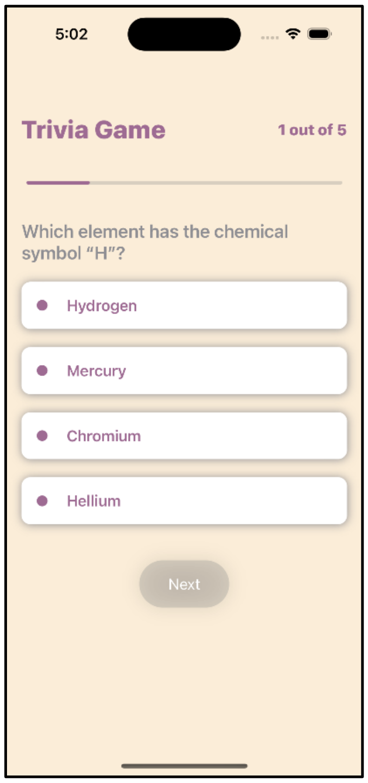


What is the capital city of Bangladesh

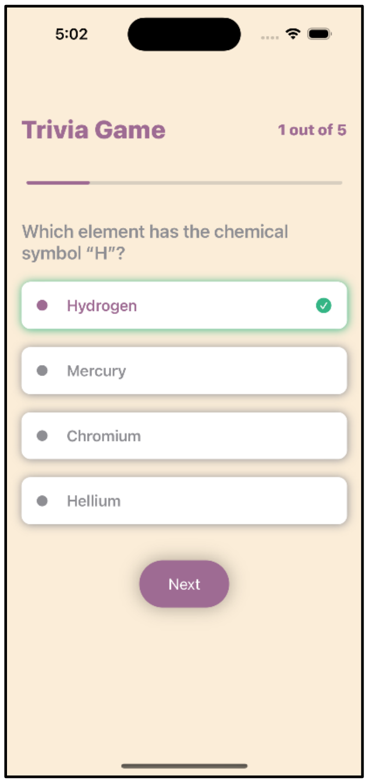
Dhaka ✓
Khulna
Rajshahi
Barisal

Quiz id: 20231127170044

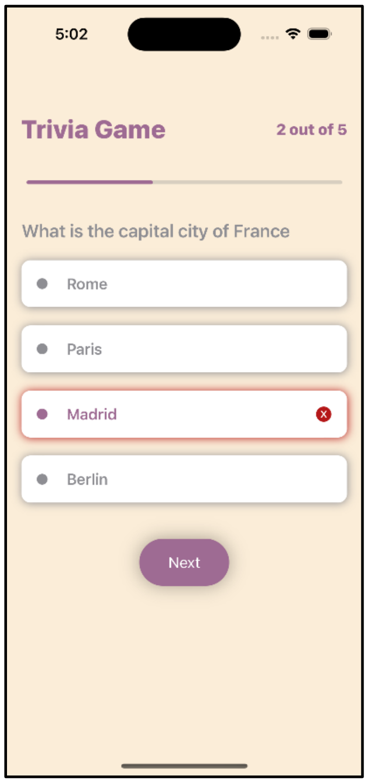




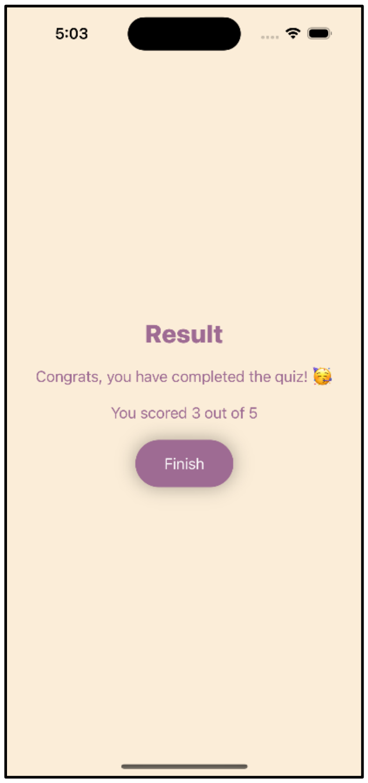
Which element has the chemical symbol “H”?



Which element has the chemical symbol “H”?

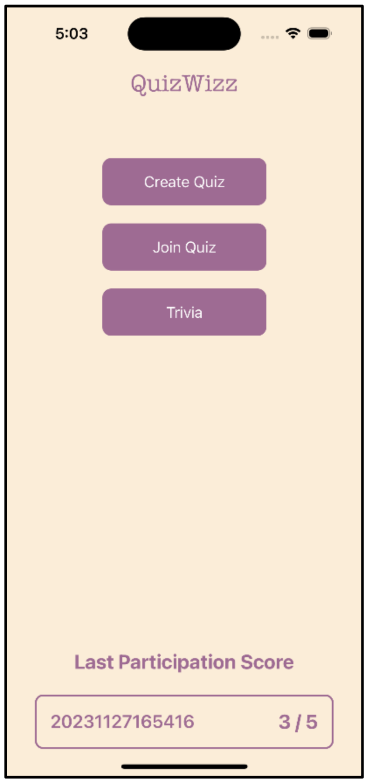


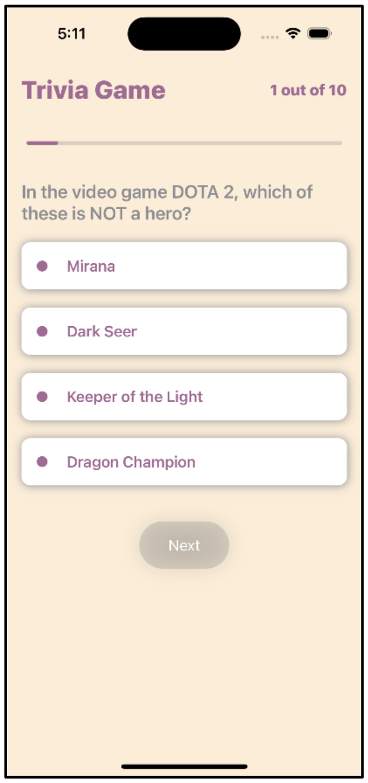
What is the capital city of France



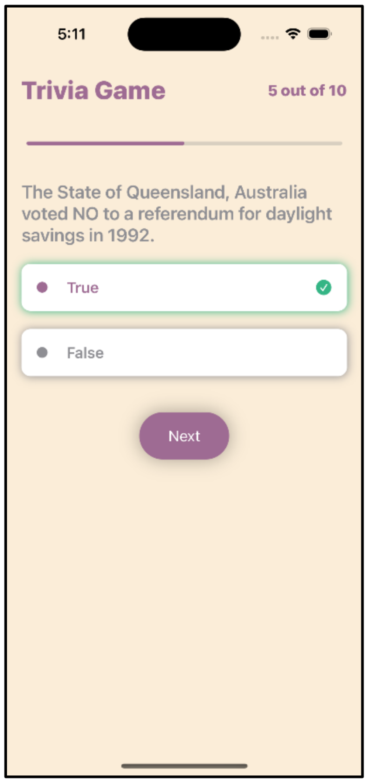
Congrats, you have completed the quiz! 🥳

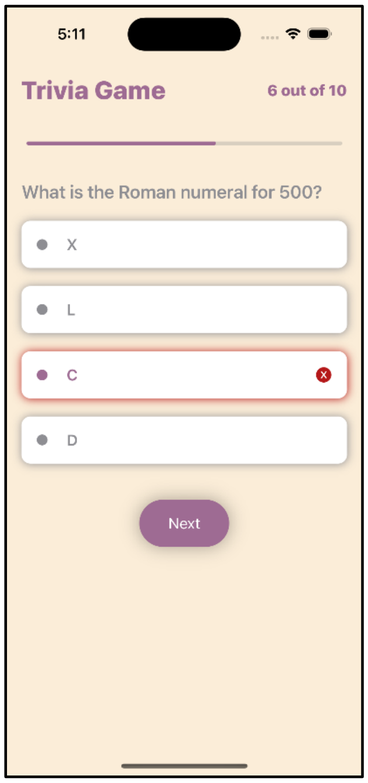
You scored 3 out of 5

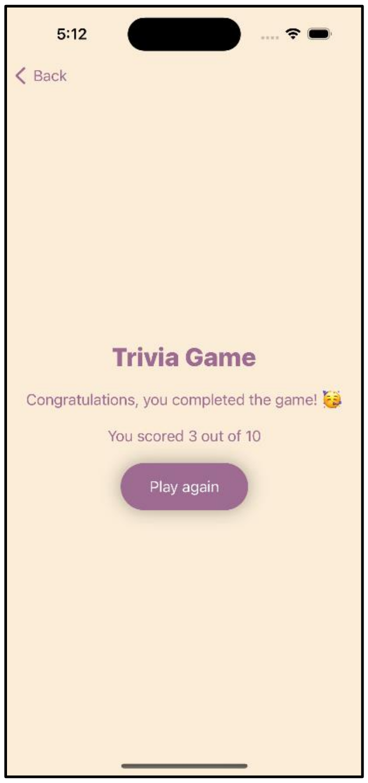




In the video game DOTA 2, which of these is NOT a hero?


5:11
Trivia Game
5 out of 10
The State of Queensland, Australia voted NO to a referendum for daylight savings in 1992.
True
False
Next





The screenshots of the app in order shows the following activities in order (Left to Right):

1. Sign in page
2. Sign up page
3. Homepage after signing in (initially)
4. Add quiz (empty fields)
5. Add quiz (fields filled)
6. Join quiz id entry page
7. Join quiz question/answer page
8. Join quiz with correct answer selected
9. Join quiz with wrong answer selected
10. Result after user defined join quiz has finished
11. Homepage after new results
12. Trivia game question/answer page
13. Trivia game with correct answer selected
14. Trivia game with wrong answer selected
15. Result after trivia game has finished

4. Discussion:

The project revolves around the development of a dynamic and engaging Quiz/Trivia App, QuizWiz using SwiftUI. Focused on providing a seamless and enjoyable user experience, the app combines user-friendly design principles with versatile features to cater to a diverse audience.

One of the objectives of the project include implementing secure user authentication, allowing users to sign in or sign up seamlessly. The app empowers users by enabling them to create and join custom quizzes, fostering a sense of ownership and community engagement. Integration of external API, Open Trivia Database, enhances the app's trivia content, ensuring a wide range of topics and challenges for users to explore.

A key aspect of the project involves establishing a comprehensive system for storing and displaying the most recently participated user defined quiz results. This feature not only allows users to track their most recent performance.

Technical optimization is a priority, with a focus on ensuring the app's performance and responsiveness. Thorough error-handling mechanisms and rigorous testing contribute to the app's stability, providing users with a reliable and satisfying experience.

In summary, the QuizWiz: A Quiz & Trivia App project is a harmonious blend of user-centric design, technical robustness, and a commitment to fostering an interactive and competitive community of users.

5. Limitations:

- ✓ Forget Password and password confirmation is not implemented
- ✓ User session is not used
- ✓ The Add Quiz option has a fixed number of options of exactly 4 answers
- ✓ The correct answer selection in Add Quiz section might cause errors in terms of multiple selection or no selection as correct answer
- ✓ The duplicated options in answer choices and questions in Add Quiz section can cause major issues as they are distinct keys in the database
- ✓ There is no distinct finish button for Add Quiz section and only uses the back button as denotation to finish adding questions
- ✓ The join quiz section ID, if entered wrong, only shows in console that the ID doesn't exist and not to the app user
- ✓ The results for user created quiz have only the option of storing the latest score and quiz id and not a complete history
- ✓ The score of the most recent user created quiz participation is shown in the homepage instead of having a separate view for it
- ✓ The results of the trivia game are not stored at all

- ✓ There is no log out system form the app once logged in
- ✓ There is no profile management section for the user to change user details
- ✓ There is no user-based customization available to the users

6. Conclusion:

The development of QuizWiz: A Quiz and Trivia App using SwiftUI has successfully materialized into a user-centric and feature-rich application. By prioritizing a seamless user experience, secure authentication, and the integration of external API for diverse content, the app offers an engaging platform for quiz enthusiasts.

7. Future Work:

The main work to do in the future in this app at first is to solve all the limitations described before. After that, we can focus on a number of new and exciting features such as:

- ✓ Enhanced Quiz Customization
- ✓ Social Integration
- ✓ Real-time Multiplayer Mode
- ✓ Gamification Elements
- ✓ Expanded Trivia Categories
- ✓ Accessibility Features
- ✓ Offline Mode and Persistent Data
- ✓ Advanced Analytics
- ✓ Dynamic Question Pooling
- ✓ Localized Content and Multilingual Support

8. References:

- ✓ https://www.swift.org/documentation/
- ✓ https://www.programiz.com/swift-programming
- ✓ https://developer.apple.com/documentation/swift