COMPUTER SCIENCE PROJECT

Author

Name-NIKET BASU
Cls-XII J ; Roll No:17
Adm No:14184/2024

Quiztastic

Prepared by: Niket Basu

1 Objective of the Proposed System

The objective of the "Quiztastic" project is to develop an interactive quiz platform that enables users to test and expand their knowledge across various subjects and difficulty levels. The system is designed to provide a seamless experience for users to register, log in, participate in quizzes, and view their performance on a leaderboard. By integrating with an external trivia API, the application offers a diverse range of questions while maintaining secure user authentication and persistent scoring via a database. This project aims to foster self-paced learning, healthy competition, and engagement through gamification.

Key Objectives:

- Automate the quiz-taking process with a user-friendly GUI.
- Ensure secure registration and login for users.
- Offer diverse quiz categories and difficulty levels.
- Store and update user scores in a persistent database.
- Motivate users through real-time feedback and leaderboards.
- Provide a scalable foundation for future educational enhancements.

2 Input and Output of the Proposed System

Inputs

- **User Details:** Username and password for registration and authentication.
- **Quiz Preferences:** Selection of quiz category, difficulty level, question type, and number of questions.
- **Quiz Responses:** User-selected answers to each quiz question.

Outputs

- **Quiz Questions:** Dynamically fetched from the Open Trivia DB API based on user preferences.
- **Immediate Feedback:** Correctness of answers after each submission.

- **Score Display:** User's score upon quiz completion.
- **Leaderboard:** Ranked list of top users based on cumulative scores.
- **Registration/Login Feedback:** Success or error messages for authentication.

Data Flow Example

1. User enters login credentials → System validates using database.
2. User selects quiz settings → System fetches relevant questions from API.
3. User answers questions → System evaluates and provides feedback.
4. Upon completion, total score is updated and displayed.
5. User can view the leaderboard at any time.

3 Functions or Features of Proposed System

Major Features:

- **User Registration & Login:**
  - Secure sign-up with hashed password storage.
  - Login authentication against SQLite database.
- **Quiz Generation:**
  - Fetches categories and questions from Open Trivia DB.
  - Customizable quiz settings (category, difficulty, type, amount).
- **Quiz Participation:**
  - Presents questions one at a time with multiple choice/boolean options.
  - Provides immediate feedback on each answer.
- **Scoring System:**
  - Tracks and updates user's cumulative score in database.
  - Displays score at quiz end.
- **Leaderboard:**
  - Shows top 10 users ranked by score.
  - Motivates with real-time competitive statistics.
- **GUI Navigation:**
  - Intuitive navigation between login, home, quiz, and leaderboard screens.
- **Error Handling & User Guidance:**

  - Informative messages for invalid inputs, API errors, and quiz completion.

Additional Features for Future Expansion:

- Biometric authentication.
- Quiz history and analytics.
- Group quizzes and challenges.

4 Front-end and Back-end to be Used

Front-end

- **Technology:** Python Tkinter (GUI library)
- **Components:**
  - Frames for each major screen (Login, Home, Quiz, Leaderboard)
  - Widgets: Labels, Buttons, Entry fields, Radiobuttons, Comboboxes, Spinboxes
  - Styling for improved user experience (fonts, colors, layouts)

Back-end

- **Database:** SQLite3
  - Stores user credentials and scores
  - Handles queries for registration, authentication, scoring, and leaderboard
- **API Interaction:** Open Trivia DB
  - Fetches quiz categories and questions remotely
- **Other Libraries:**
  - `requests` for making HTTP requests to API.
  - `hashlib` for password hashing.
  - `random` for shuffling answer options.
  - `html` for handling special characters in questions and answers.

Architecture Overview:
The GUI interacts with the back-end database and the trivia API, ensuring smooth data flow and real-time user feedback.

5 Hardware & Software to be Used

Hardware Requirements

- Standard personal computer or laptop
- Minimum 2GB RAM, recommended 4GB+
- No specialized hardware required (optional biometric device for future)

Software Requirements

- **Operating System:** Windows/Linux/Mac OS
- **Programming Language:** Python 3.x
- **Libraries:**
  - Tkinter (built-in with Python)
  - requests (`pip install requests`)
  - SQLite3 (built-in with Python)
  - hashlib, random, html (standard library)
- **Additional Tools:**
  - Internet connection (for API access and initial setup)
  - IDE/Text Editor (e.g., Visual Studio Code, PyCharm, IDLE)

6 Scope and Limitations of the Project

Scope

- Provides an engaging platform for individual users to participate in quizzes.
- Suitable for educational institutions, self-learners, and competitive environments.
- Easily extendable for new features (group quizzes, analytics, additional question sources)
- Promotes gamification and learning retention.

Limitations

- **Single-device focus:** No native support for mobile/web platforms.
- **API dependency:** Quiz content relies on external API availability and quality.
- **Database scalability:** SQLite is ideal for small to moderate user bases but not for large-scale deployments.

- **Limited user analytics:** No detailed tracking of user progress or question statistics.
- **No offline quiz:** Requires internet for fetching new questions.

Conclusion

Quiztastic is a robust and scalable quiz platform that leverages Python, Tkinter, and SQLite to deliver a dynamic and interactive user experience. The system supports secure user management, customizable quizzes, real-time scoring, and competitive leaderboards. With clear design and modular architecture, it offers a foundation for future enhancement and adaptation to broader educational needs.