Report: Python MCQ Application Project

Introduction

This project is a quiz app for computer science students. It lets users take multiple-choice tests, tracks their scores, and saves their history. The app runs in the console, making it easy to use without a fancy interface. Admins can add new questions, and users can pick topics like Python or Cybersecurity.

How It Works

- **User Management**: When you start the app, you type a username. If you're new, it creates a profile. Returning users see their past scores and test dates. All this is saved in a JSON file.
- **Questions & Answers**: Questions are stored in a JSON file, split into categories. Each question has three options, and the app checks if your answer is correct.
- **Taking the Quiz**: You pick a category (or choose "All"), answer questions one by one, and get immediate feedback. After finishing, your score is saved.
- **Admin Features**: Admins (like "riad" or "foued") can add new questions to any category.

Technical Stuff

- **JSON Files**: Used for storing users and questions because it's simple and works well with Python.
- **Classes & Functions**: The main class MCQApplication handles everything—loading data, running quizzes, saving results. Functions are split into smaller tasks (e.g., load_users(), run_test()) to keep the code clean.
- **Input Validation**: The app forces you to type valid answers (a/b/c) and checks if usernames or categories exist.

Python MCQ Application Project Report

Intro

This app is basically a quiz tool for comp sci students. You answer multiple-choice questions, get your score, and see how you've done over time. No fancy graphics—just a console interface that gets straight to the point. Admins can add new questions, and everyone can pick topics like Python or Cybersecurity.

How It Works (No Robot Talk, Promise)

1. **Starting Up**
   - Open the app, and it's like, *"Yo, what's your username?"*
   - If you've used it before, it digs up your past scores and test dates from a JSON file (think of it as a digital notebook that never forgets).
   - New user? No biggie. It creates a profile for you on the spot.
2. **Taking the Quiz**
   - Pick a category (like "Python" or "Algorithms") or go full chaos mode with *"All Categories."*
   - The app grabs questions from the JSON file, shuffles them (so you don't memorize the order), and fires them at you one by one.

    - Each question has three options (**a**, **b**, **c**). If you try typing something dumb like “z” or “🍕,” it’ll keep bugging you until you pick a real answer.
3. **Feedback & Scores**
    - After each question, it instantly tells you if you nailed it or faceplanted. If you messed up, it shows the correct answer so you can learn.
    - At the end, it slaps your final score on the screen (*“14/20! Almost there!”*) and saves it to your profile.
4. **Admin Stuff**
    - Admins (like “riad” or “foued”) get a secret menu to add new questions. They pick a category, type the question + options, and tag the right answer. The app tucks it into the JSON file so everyone else gets to sweat over it later.
5. **Extras**
    - **Export Scores**: You can dump your entire quiz history into a CSV file (aka a spreadsheet for nerds). Handy for bragging rights or tracking progress.
    - **History Tracking**: Every test you take gets logged with the date, category, and score. It’s like a workout log but for your brain.

Under the Hood (But Keepin’ It Simple)

- **JSON Files**: Stores users and questions in these files because they’re easy to read/write in Python. No complicated databases here.
- **Random Questions**: Uses random.sample() to pick questions without repeats. No one wants the same quiz twice.
- **Input Checks**: The app’s picky. It won’t let you type nonsense answers or fake categories.
- **Classes & Functions**: The main class MCQApplication does the heavy lifting—loading data, running quizzes, saving results. Code’s split into smaller chunks (like load_users() or run_test()) to keep things tidy.

Stuff That Went Wrong (And How We Fixed It)

1. **Missing Files**: If the JSON files got deleted, the app would crash. Fixed it by adding error messages like *“Hey, where’d the questions go?!”* and checking if files exist.
2. **Broken History**: Sometimes the app forgot to save scores. Solved by updating the user’s profile immediately after each test.
3. **Admin Drama**: Hardcoding admin usernames (like “riad”) wasn’t secure, but it works for a school project. Maybe upgrade later.
4. **Category Glitches**: If you typed a category that didn’t exist, the app would freak out. Added checks to say *“Nope, that category’s not real”* before starting the quiz.

Cool Features

- **Category Selection**: Pick your poison—Python, AI, Cybersecurity, etc.
- **Instant Feedback**: No waiting to see if you’re right or wrong.
- **CSV Export**: For folks who love spreadsheets.
- **Admin Mode**: Add new questions without touching the code.

Missing Pieces

- **Timer**: No pressure mode yet. Could add a countdown later for extra stress.
- **More Question Types**: Stuck with three-option questions. Maybe add true/false or images someday.

Testing (We Tried to Break It)

- **Fake Users**: Made accounts like “QuizMaster3000” and “Noob123” to see if scores saved.
- **Invalid Inputs**: Typed “potato” as an answer. The app just yelled *“Nope, pick a/b/c!”*
- **Edge Cases**: Took quizzes with all correct/wrong answers. The app didn’t explode (thankfully).
- **CSV Checks**: Opened exported files in Excel to make sure dates and scores lined up.

Final Thoughts

This app does what it needs to: lets students grind through MCQs and track their progress. The code’s not perfect (looking at you, hardcoded admins), but it’s a solid foundation. Future upgrades? A timer, more question types, or even a leaderboard could spice things up. For now, it’s a no-nonsense tool that gets the job done.

1. **File Errors**: If the JSON files are missing or messed up, the app crashes. Fixed this by adding error messages and checks for file existence.
2. **Random Questions**: Selecting random questions from a category was tricky. Used Python’s random.sample() to grab a subset without repeats.
3. **Saving History**: Initially, the app didn’t update user history properly. Solved by appending results to the user’s list and saving the file immediately after each test.
4. **Admin Access**: Hardcoded admin usernames (like “riad”) into the code. Not the most secure, but works for a school project.

Features Added

- User login/history tracking.
- Category selection and random question pulls.
- Score feedback with correct/incorrect answers.
- Export results to CSV.
- Admin question adding.

What’s Missing

- **Timer**: The optional timer for questions wasn’t added yet.
- **More Question Types**: Only supports three-option questions.

Testing

Tested by creating fake users, taking quizzes in different categories, and checking if scores saved correctly. Also tried breaking the app (typing invalid answers, using wrong categories) to see if error messages worked. Exported CSV files to confirm data looked right.

Conclusion

The app does what it’s supposed to: lets students practice MCQs and track progress. The code isn’t perfect (e.g., hardcoded admins), but it’s a solid start. For future work, adding a timer or more question formats would make it even better.