People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of Science and Technology Houari Boumediene
Faculty of Computer Science

Advanced Programming Project Report: "DevQuiz"

Development of a Quiz Application for Students in Python

Team: SheCodes

Team Members:

Ghribi Ahlam
Laidli Besmala Meriem
Mehanni May Fatema Zohra

Academic Year: 2024/2025

Contents

Chapter 1

Introduction

DevQuiz is an interactive quiz application designed to provide students with an engaging way to test their knowledge across various domains. The application features both a command-line interface (CLI) and a web-based frontend powered by Django.

Objective of the Project:

- Develop an intuitive and user-friendly quiz platform.
- Offer features for quiz creation, user performance tracking, and feedback.
- Support data management using JSON and CSV for scalability and flexibility.
- Provide a robust architecture to handle both terminal-based and web-based user interactions.

Main Features:

- **Quiz Creation and Management:** Users can manage quizzes with questions stored in JSON files.
- **Dual Interfaces:**
  - Command-Line Interface (CLI) for simple and fast interactions.
  - Web-based UI using Django for enhanced user experience.
- **Performance Tracking:** User scores, percentages, and timestamps are saved in a CSV file for detailed analysis.
- **Categorized Quizzes:** Users can filter questions based on categories like Python, Java, HTML, JavaScript, CSS, PHP, and SQL.
- **Real-Time Feedback:** Results, including correct answers, are displayed after each quiz.

Chapter 2

Technologies Used

2.1 Command-Line Execution

- **Python:** Core language for the CLI application.
- **Libraries:**
  - `json`: For handling quiz data.
  - `csv`: For exporting user history and scores.
  - `os`: For file system operations.
  - `datetime`: For managing timestamps.

2.2 The Web App

- **Django Framework:** Used for the MVC structure to manage the web application.
- **HTML, CSS, and JavaScript:** For creating a responsive and interactive frontend.
- **Libraries:**
  - `django.shortcuts`: For rendering templates and handling redirects.
  - `django.http`: For HTTP responses and data communication.
  - `json`: For data serialization.
  - `csv`: For exporting data to user history files.

Chapter 3

Project Architecture

The project architecture is divided into two parts:

3.1 Command-Line Execution

The CLI-based quiz application consists of:

- `main.py`**:** The main script for running the quiz in the terminal.
- **Data Files:**
  - `questions.json`: Stores quiz questions.
  - `users.json`: Maintains user information.
  - `userhistory.csv`: Logs user scores and timestamps.

  The directory structure for the CLI version is:

```
/
 main.py
 questions.json
 users.json
 user_history.csv
```

3.2 Web Application

The web-based application follows Django's MVC pattern:

- **Models (**`models.py`**):** Represent the database schema for users and quiz data.
- **Views (**`views.py`**):** Contain the logic for rendering quiz pages and handling user requests.
- **Templates (**`templates/`**):** HTML files for UI rendering.
- **Static Files (**`static/`**):** CSS and JavaScript files.

  The Django project structure is:

```
qcm_project/
 manage.py
 db.sqlite3
 qcm_project/
    settings.py
    urls.py
    ...
 qcm_app/
    models.py
    views.py
    templates/
       index.html
       quiz.html
       ...
    static/
        css/
        js/
 data/
```

Figure 3.1: DevQuiz Logo

Chapter 4

Development Process

4.1 Phase 1: Design Phase

UI/UX Design: The design for the DevQuiz application was crafted using **Figma** to ensure that the application is intuitive, user-friendly, and responsive across different devices. The primary goal was to create a simple, modern, and accessible interface for users to interact with the quiz.

4.2 Phase 2: Development Phase

The **development phase** began with the implementation of the core features of the DevQuiz application. This included both the **command-line interface (CLI)** and the **web application** developed using **Django**.

User Interface Development:

- The home page was implemented as designed in Figma, with attention to ensuring smooth interaction.
- Input fields and buttons were integrated to allow users to submit their usernames and begin the quiz.

Feature Integration:

- **User Authentication:** A simple form to input a username was created, followed by saving that information in session variables for personalized experiences.
- **Quiz Mechanics:** A quiz engine was created to load questions from the backend (using JSON files), and users could submit their answers to calculate their scores.

```
Welcome to DevQuiz! Test, Learn, and Conquer!
Enter your username please: ahlam
Welcome back, ahlam!

Your Quiz History:
Date: 2024-12-21 20:59:44
Score: 17/20
Category: Python

Date: 2025-01-17 16:42:07
Score: 8.6/20
Category: SQL

Date: 2025-01-17 17:43:27.301724
Score: 8.6/20
Category: PHP


DevQuiz Menu:
1. Take Quiz
2. View History
3. View Best Users
4. Exit
Enter your choice (1-4): 3

Top Performers:
1. May: 18.5/20 average
2. Besmala: 18.0/20 average
3. Ahlam: 16.7/20 average
4. Sarah: 16.5/20 average
5. Adam: 15.5/20 average

DevQuiz Menu:
1. Take Quiz
2. View History
3. View Best Users
4. Exit
```

Figure 4.1: CMD EXECUTION

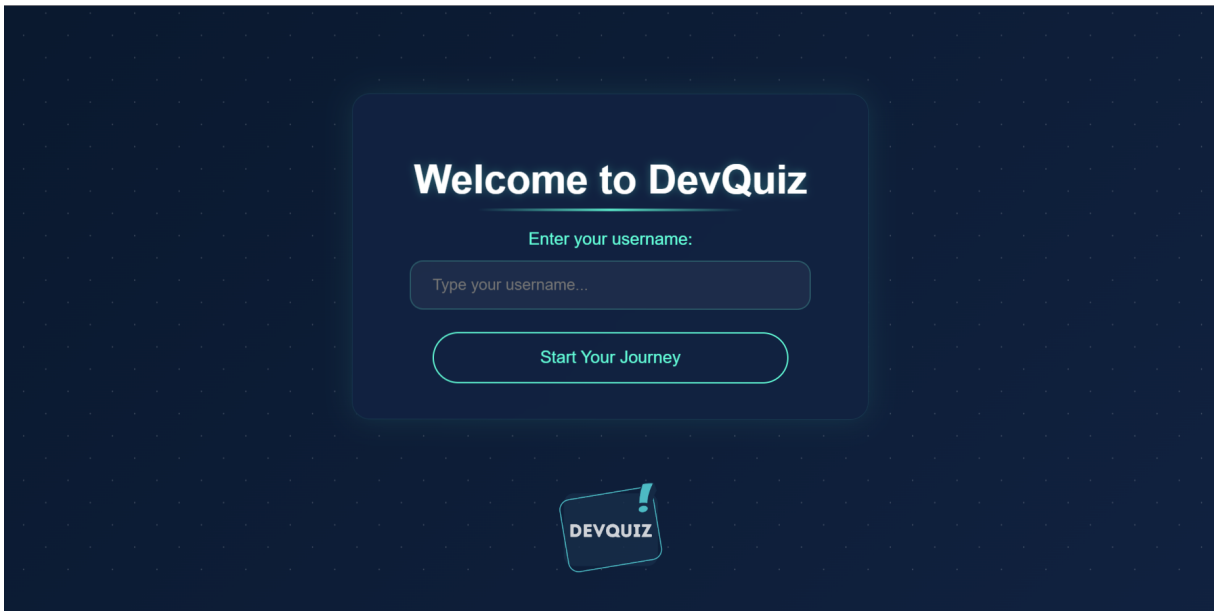

Figure 4.2: WEB APP HOME PAGE

Chapter 5

Challenges Encountered

Throughout the project, several challenges were encountered and successfully resolved:

5.1 Integration of the Python CLI with Django Web Application

Problem: The primary challenge was ensuring seamless communication between the command-line interface (CLI) and the Django-based web application, as they were developed separately. **Solution:** To resolve this, we ensured that both versions could share the same data files (e.g., JSON for quizzes and CSV for performance tracking). We also implemented a simple API or file-based communication to allow user data to be passed between both interfaces.

5.2 Managing Data Across Multiple Platforms

Problem: Storing and retrieving quiz questions, user answers, and performance data across both the CLI and web platforms presented a challenge. **Solution:** We standardized the data format by using JSON for quiz questions and CSV for user scores, which made it easy to handle across both interfaces. This approach allowed for simple file handling and ensured data consistency.

5.3 Ensuring an Intuitive User Interface

Problem: Designing a user interface that is both intuitive and interactive for a broad range of users was challenging, especially when considering both a CLI and a web application. **Solution:** We focused on creating a user-friendly web interface with clean layouts and simple navigation. For the CLI, we ensured that the quiz was easy to follow and provided feedback on the user's progress after each question.

Chapter 6

Conclusion

Thanks to its intuitive interface and polished design, the **DevQuiz** application offers users a unique and enjoyable experience. Whether on the terminal or the web, every user can easily navigate through the different stages of the quiz while tracking their progress in real-time. The project was designed to foster interactive learning, and we are proud to see it fulfilling that goal. This work represents the collective effort and enthusiasm of our team to provide a dynamic, simple, and motivating platform for students and programming enthusiasts alike.

Developing **DevQuiz** has allowed us to overcome technical challenges while also refining our skills in design, programming, and testing. We hope this application will inspire and assist anyone looking to enhance their knowledge while having fun. With **DevQuiz**, learning becomes an exciting journey full of discoveries!