Quiz App Documentation

Ankala Santhkumar

July 26, 2025

Contents

1 Introduction

The Quiz App is a web-based application designed to assess users' knowledge across three domains: Programming, Aptitude, and English Grammar. It features a user-friendly interface with a timed, multi-section quiz format, secure login, and detailed results reporting. The app is developed with a static HTML frontend (`quiz.html`) and a Spring Boot backend, deployed on GitHub Pages for the frontend and DigitalOcean or Heroku for the backend. This documentation provides a comprehensive guide to the project's features, architecture, setup, deployment, usage, and troubleshooting.

1.1 Purpose

The Quiz App serves as an educational tool for evaluating skills in technical and non-technical areas. It is designed for individual users, with features like timed sections, keyboard navigation, and a pass/fail system based on a 70% threshold.

1.2 Scope

The app includes:

- A login system with predefined credentials.
- Three quiz sections, each with 10 questions.
- A 5-minute timer per section with auto-submission.
- Score display only on the results page.
- Deployment on free or low-cost platforms.

Exclusions: Database storage, user registration, or advanced analytics.

2 Features

The Quiz App offers the following features:

2.1 Login

- Credentials: User ID `santhkumar`, Password `santhkumar`.
- Blinking credential display for user guidance.
- Client-side validation with error message for invalid inputs.

2.2 Instructions

- Displays quiz rules and structure.
- 30-second countdown before auto-starting the quiz.
- Manual "Next" button to proceed early.

2.3 Quiz Sections

- **Sections**: Programming (Java/SQL), Aptitude, English Grammar.
- **Questions**: 10 per section, multiple-choice with four options.
- **Timer**: 5 minutes per section, with a warning at 30 seconds remaining.
- **Navigation**:
  - "Next Section" for Sections 1 and 2.
  - "Submit" for Section 3.
  - "Submit and Exit" available in all sections with confirmation dialog.
- **Keyboard Navigation**: Arrow keys to navigate options, Enter to select.
- **Score**: Hidden during the quiz, displayed only on results page.

2.4 Results

- Per-section scores (e.g., Programming: 7/10).
- Final score (e.g., 22/30, 73.33%).
- Pass status: Pass if $\geq$ 70% (21/30 correct).
- Copyright notice: © 2025 Ankala Santhkumar.

3 Architecture

The Quiz App is a client-server application with a static frontend and a RESTful backend.

3.1 Frontend

- **File**: `quiz.html` (HTML, CSS, JavaScript).
- **Location**: `src/main/resources/static/quiz.html`.
- **Functionality**:
  - Renders login, instructions, quiz, and results pages.
  - Uses CSS for responsive design (mobile, tablet, desktop).
  - JavaScript handles UI interactions, timers, and API calls.
- **Key Elements**:
  - Login form with blinking credentials.
  - Progress bar for section tracking.
  - Hidden score element (`<p id="score">`).

3.2 Backend

- **Framework**: Spring Boot (Java).
- **Controller**: `QuestionController.java` in `src/main/java/com/example/quizapp/control`
- **API Endpoints**:
  - `GET /question/section/{section}`: Returns 10 questions for the specified section (Programming, Aptitude, English Grammar).
  - `POST /question/submit?questionId={id}&answer={answer}`: Evaluates an answer, returns "Correct!" or "Incorrect!".
- **Configuration**: `application.properties` sets port 8080.

3.3 Data Flow

1. User logs in via `quiz.html`.
2. Frontend fetches questions from `GET /question/section/{section}`.
3. User submits answers via `POST /question/submit`.
4. Backend evaluates answers, frontend displays results.

4 Setup Instructions

To run the Quiz App locally, follow these steps:

4.1 Prerequisites

- **Java**: JDK 17 or later (JDK 24 tested).
- **Maven**: Version 3.8+.
- **Git**: For version control.
- **Node.js**: Optional for development tools.
- **Operating System**: Windows, macOS, or Linux.

4.2 Cloning the Repository

1. Install Git: https://git-scm.com.
2. Clone the repository:

```
cd C:\Users\ankal\Downloads
git clone https://github.com/<username>/quizapp.git
cd quizapp
```

4.3 Project Structure

- `src/main/java/com/example/quizapp/controller/QuestionController.java`: Backend logic.
- `src/main/resources/static/quiz.html`: Frontend.
- `src/main/resources/application.properties`: Configuration.
- `pom.xml`: Maven dependencies.

4.4 Building and Running

1. Navigate to project directory:

```
cd C:\Users\ankal\Downloads\quizapp
```

2. Build the project:

```
mvn clean install
```

3. Run the application:

```
mvn spring-boot:run
```

4. If Java 24 warnings occur, use:

```
mvn spring-boot:run -DargLine="--enable-native-access=ALL-UNNAMED"
```

5. Access `http://localhost:8080/quiz.html` in a browser.

4.5 Testing Locally

- **Login**: Use `santhkumar/santhkumar`.
- **Instructions**: Verify 30-second countdown.
- **Quiz**: Check question loading, timers, navigation.
- **Results**: Ensure score is hidden until results page.
- **Console**: Open browser DevTools (F12), check logs.

5 Deployment Instructions

The Quiz App is deployed with the frontend on GitHub Pages and the backend on DigitalOcean or Heroku.

5.1 Frontend: GitHub Pages

1. **Push to GitHub**:

```
cd C:\Users\ankal\Downloads\quizapp
git add .
git commit -m "Initial quiz app commit"
git push origin main
```

2. **Create index.html**:

```
copy src\main\resources\static\quiz.html index.html
git add index.html
git commit -m "Add index.html for GitHub Pages"
git push origin main
```

3. **Enable GitHub Pages**:
   - Go to https://github.com/<username>/quizapp > Settings > Pages.
   - Source: Deploy from a branch.
   - Branch: `main`, Folder: `/root`.
   - Click Save (if disabled, see Troubleshooting).
4. **Access Site**: Visit https://<username>.github.io/quizapp after 1–10 minutes.

5.2 Backend: DigitalOcean

1. **Sign Up**: Register at https://digitalocean.com.
2. **Create App**:
   - Dashboard > Create > Apps.
   - Connect GitHub, select `quizapp`.
   - Resource Type: Web Service, Port: 8080.
   - Deploy.
3. **Get URL**: Note app URL (e.g., https://your-app-12345.ondigitalocean.app).
4. **Update index.html**:

```
fetch('https://your-app-12345.ondigitalocean.app/question/section/${
    encodeURIComponent(section)}')
fetch('https://your-app-12345.ondigitalocean.app/question/submit?
    questionId=${questionId}&answer=${encodeURIComponent(answer)}', {
    method: 'POST' })
```

```
git add index.html
git commit -m "Update backend URL"
git push origin main
```

5.3 Alternative Backend: Heroku

1. **Install Heroku CLI**: https://devcenter.heroku.com/articles/heroku-cli.
2. **Deploy**:

```
cd C:\Users\ankal\Downloads\quizapp
heroku login
heroku create quizapp-backend
heroku git:remote -a quizapp-backend
git push heroku main
```

3. **Update index.html**:

```
fetch('https://quizapp-backend.herokuapp.com/question/section/${
    encodeURIComponent(section)}')
fetch('https://quizapp-backend.herokuapp.com/question/submit?
    questionId=${questionId}&answer=${encodeURIComponent(answer)}', {
    method: 'POST' })
```

```
git add index.html
git commit -m "Update backend URL for Heroku"
git push origin main
```

5.4 Custom Domain (Optional)

1. Purchase a domain (e.g., `yourquizapp.com`).
2. Configure DNS:
   - CNAME: `www` → `<username>.github.io`.
   - A records:

     ```
     185.199.108.153
     185.199.109.153
     185.199.110.153
     185.199.111.153
     ```

3. Add CNAME file:

```
echo "yourquizapp.com" > CNAME
git add CNAME
git commit -m "Add custom domain"
git push origin main
```

4. Configure GitHub Pages: Settings > Pages > Custom domain > `yourquizapp.com`.

6 Usage Guide

1. **Access**: Visit https://<username>.github.io/quizapp.
2. **Login**: Enter `santhkumar/santhkumar`.
3. **Instructions**: Read rules, wait 30 seconds or click "Next".
4. **Quiz**:

- Answer 10 questions per section within 5 minutes.
- Use “Next Section” (Sections 1–2), “Submit” (Section 3), or “Submit and Exit”.
- Navigate options with arrow keys, select with Enter.

5. **Results**: View per-section scores, final score, and pass status.

7 Troubleshooting

- **GitHub Pages Save Button Disabled**:
  - Ensure `index.html` is in repository root.
  - Make repository public (Settings > General).
  - Verify admin permissions.
  - Use GitHub Actions:

```
mkdir -p .github/workflows
```

Create `pages.yml`:

```
name: Deploy GitHub Pages
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Pages
        uses: actions/configure-pages@v3
      - name: Upload artifact
        uses: actions/upload-pages-artifact@v2
        with:
          path: .
      - name: Deploy to GitHub Pages
        id: deployment
        uses: actions/deploy-pages@v2
```

```
git add .github/workflows/pages.yml
git commit -m "Add GitHub Actions for Pages"
git push origin main
```

- **Questions Not Loading**:
  - Test backend: https://your-app-12345.ondigitalocean.app/question/section/Programming.
  - Check `index.html` fetch URLs.
  - Add CORS to `QuestionController.java`:

```
import org.springframework.web.bind.annotation.CrossOrigin;
@RestController
@CrossOrigin(origins = "https://<username>.github.io")
public class QuestionController { ... }
```

- **Backend Errors**:
  - DigitalOcean: Check logs in App Platform.
  - Heroku: Run `heroku logs -tail`.
- **Contact Support**:
  - GitHub: https://support.github.com.
  - DigitalOcean: Dashboard tickets.
  - Heroku: https://help.heroku.com.

8 Future Enhancements

- **Database Integration**: Use PostgreSQL for question storage.
- **User Accounts**: Add registration and authentication.
- **Logout Button**: Return to login page.
- **Results API**: Store and retrieve past scores.
- **Accessibility**: Enhance ARIA labels and screen reader support.