Project Name: EduScore – A simple yet powerful name emphasizing education and scoring.

Abstract

The **Online Assessment System** is a user-friendly platform designed for Aroma Brand Solutions' EdTech product to simplify the process of conducting MCQ-based tests online. The system allows users to log in with predefined credentials, take a timed test of 25 questions, and view their results instantly after submission.

The platform ensures security by disabling inspect mode and refreshing the test if the user tries to leave the browser. All user data, including scores and test details, are securely stored in Google Sheets. After completing the test, users can download a PDF report with their results, including their score, test date, and credentials.

This project aims to provide a simple, secure, and efficient way to conduct assessments, helping educators and learners save time and effort. With a focus on usability and reliability, it's a practical solution for modern educational needs.

Core Functionalities to Implement

1. **User Authentication**: Validate predefined usernames and passwords securely.
2. **Dynamic Question Delivery**: Randomly select and present 25 questions from a pool of 150+ MCQs based on difficulty.
3. **Test Timer and Session Control**:
    - 20-minute timer auto-submits the test after completion.
    - If the user leaves the browser, reload and reset the session.
4. **Result Display and Download**:
    - Compute and show the result immediately after submission.
    - Allow the user to download a result PDF containing their name, score, and test date.
5. **Database Integration**: Store user credentials, test results, and MCQs.

Week 1: Core Development

Day 1-2: Planning and Database Setup

Goals:

1. Finalize system requirements and workflows.
2. Set up Google Sheets for data storage.
3. Establish a secure connection using the Google Sheets API.

Tasks:

- Define workflows:
  - Login and authentication.
  - Question fetching and test flow.
  - Result storage and PDF generation.
- Create Google Sheets:
  - **Sheet 1**: users (columns: Username, Password (hashed), Last Test Date, Score).
  - **Sheet 2**: questions (columns: Question, Options, Correct Answer, Difficulty).
  - **Sheet 3**: results (columns: Username, Test Date, Score).
- Test API connection to Google Sheets using OAuth.

Day 3-4: Backend Development

Goals:

1. Implement APIs for authentication, question fetching, and result submission.
2. Create logic for selecting 25 questions by difficulty.

Tasks:

- **User Authentication**:
  - Check credentials against the users sheet.
  - Hash passwords for security.
- **Question API**:
  - Fetch 25 random questions from the questions sheet.
  - Ensure difficulty-based selection (e.g., 10 easy, 10 medium, 5 hard).
- **Result API**:
  - Update results sheet after test submission.
  - Include timestamp and score calculation logic.
- Create a test script to validate API endpoints.

Day 5-6: Frontend Development

Goals:

1. Build responsive UI for login, test, and result pages.
2. Integrate frontend with backend APIs.

Tasks:

- **Login Page**:
  - Form for username and password.
  - On success, redirect to the test page.
- **Test Page**:
  - Display one question at a time.
  - Navigation controls (Next, Previous).
  - Countdown timer for 20 minutes.
  - Alert on browser leave and session reload.
- **Result Page**:
  - Display score, test date, and a "Download PDF" button.

Week 2: Testing, Optimization, and Deployment

Day 7-8: Feature Finalization

Goals:

1. Add security measures.
2. Implement PDF report generation.

Tasks:

- **Security**:
  - Disable inspect mode, right-click, and shortcuts (e.g., Ctrl+Shift+I).
  - Add focus/blur event tracking for tab switching.
- **PDF Report**:
  - Use jsPDF or PDFMake to generate downloadable reports.
  - Include: Username, Test Date, Score, and list of attempted questions.

Day 9-10: Integration Testing

Goals:

1. Test all system functionalities.
2. Fix bugs and optimize performance.

Tasks:

- Test scenarios:
  - Login failures and success.
  - Question selection and navigation.
  - Timer accuracy and session reload.
  - Result storage and PDF download.
- Fix bugs and improve API response times.

Day 11-12: Deployment

Goals:

1. Host the system online.
2. Conduct final testing in the live environment.

Tasks:

- Deploy frontend on GitHub Pages, Netlify, or Vercel.
- Deploy backend using Google Apps Script or a lightweight Node.js server.
- Set environment variables for secure API key storage.
- Perform end-to-end testing.

Day 13-14: Final Adjustments

Goals:

1. Deliver the final system.
2. Provide documentation for future maintenance.

Tasks:

- Create a user manual for administrators.
- Document APIs and code for easy updates.
- Conduct a walkthrough with stakeholders.

Key Deliverables

1. **Week 1**:
    - Functional APIs for user authentication, question fetching, and result submission.
    - Basic frontend for login, test, and result pages.
    - Google Sheets database setup and integration.
2. **Week 2**:
    - Fully functional and secure frontend-backend integration.
    - PDF generation and download.
    - Deployment and final testing.

Team Workflow

- **Team A**: Backend Development (APIs for login, questions, results).
- **Team B**: Frontend Development (UI, test logic, timer).
- **Team C**: Security and PDF Integration (inspect mode disable, jsPDF setup).
- **Team D**: Testing and Deployment (QA, bug fixes, hosting).

This roadmap ensures a complete system within 2 weeks by working efficiently and in parallel. Let me know if you need further elaboration on any section!