PROBLEM STATEMENT FOR BUILDX

PROBLEM STATEMENT 1

Advanced E-Commerce Cart System

Problem Statement:

Design and develop a fully functional e-commerce frontend application that simulates real-world online shopping functionality using HTML, CSS, and JavaScript.

Objective:

To evaluate the participant's ability to manage dynamic UI rendering, handle cart state management, perform price calculations, and implement persistent data storage.

Requirements:

- Display a list of products with details (name, price, image, category).
- Implement Add to Cart functionality.
- Allow users to remove products from the cart.
- Enable quantity update for each product.
- Automatically calculate total price based on cart contents.
- Implement product search functionality.
- Implement category filtering.
- Cart data must persist using localStorage or Firebase.

Expected Output:

A responsive e-commerce website where users can browse products, manage a cart dynamically, and retain cart data even after page refresh.

PROBLEM STATEMENT 2

Dynamic Quiz Builder and Evaluator

Problem Statement:

Develop a web application that allows dynamic creation and participation in quizzes with automated evaluation.

Objective:

To assess dynamic DOM manipulation, logical handling of question structures, and implementation of score computation with persistent storage.

Requirements:

- Provide functionality to add quiz questions dynamically.
- Display quiz interface for user participation.
- Automatically calculate and display score after submission.
- Show performance summary message.
- Store quiz history using localStorage or Firebase.
- Optional: Timer feature for quiz attempt.

Expected Output:
A dynamic quiz platform where quizzes can be created, attempted, evaluated, and stored for future reference.

PROBLEM STATEMENT 3

Smart Quiz Platform with Timer and Leaderboard

Problem Statement:
Create an advanced quiz system with predefined questions, timed responses, and a leaderboard mechanism.

Objective:
To evaluate handling of timers, real-time score updates, and ranking logic using persistent data storage.

Requirements:

- Display predefined multiple-choice questions.
- Implement timer per question or overall quiz timer.
- Automatically submit quiz when timer ends.
- Calculate and display final score.
- Maintain leaderboard based on stored scores.
- Use localStorage or Firebase for storing leaderboard data.

Expected Output:
A fully functional timed quiz application with score calculation and leaderboard display.

PROBLEM STATEMENT 4

Student Performance Analytics Dashboard

Problem Statement:
Develop a dashboard to manage and analyze student academic performance data.

Objective:
To test data structuring, calculation logic, and dashboard-style UI presentation.

Requirements:

- Add multiple students.
- Enter marks for at least five subjects.
- Calculate average marks.
- Assign grade based on defined criteria.
- Display class average.
- Identify top performer automatically.
- Persist student data using localStorage or Firebase.

Expected Output:
An interactive dashboard displaying individual and class performance metrics with persistent student records.

PROBLEM STATEMENT 5

Health and Fitness Progress Analyzer

Problem Statement:

Design a web-based health tracking system to monitor fitness progress over time.

Objective:

To evaluate logical implementation of health metrics calculation and progress visualization.

Requirements:

- Input daily weight.
- Calculate BMI based on height and weight.
- Allow user to set target weight.
- Display weight progress graph.
- Predict approximate goal achievement timeline (simple logic).
- Store health records persistently.

Expected Output:

A health tracking system with progress visualization and long-term data retention.

PROBLEM STATEMENT 6

Smart Attendance Tracking System

Problem Statement:

Create an attendance management system to record and analyze student attendance data.

Objective:

To test structured data handling, percentage calculations, and conditional alert mechanisms.

Requirements:

- Add student names.
- Mark daily attendance (Present/Absent).
- Calculate attendance percentage automatically.
- Display warning for low attendance (below threshold).
- Provide monthly summary.
- Persist attendance records.

Expected Output:

A system that maintains attendance data and generates percentage analysis with alerts.

PROBLEM STATEMENT 7

Workout Planner with Weekly Schedule

Problem Statement:

Develop a workout planning system to organize and monitor weekly fitness routines.

Objective:

To evaluate scheduling logic, completion tracking, and data persistence.

Requirements:

- Add exercises.
- Assign exercises to specific days of the week.
- Mark workouts as completed.
- Calculate weekly completion percentage.
- Persist weekly workout schedule.

Expected Output:
A weekly workout planner with completion tracking and stored schedule data.

PROBLEM STATEMENT 8

Personal Knowledge Base Manager

Problem Statement:
Create a web application for managing personal notes and categorized knowledge entries.

Objective:
To assess CRUD operations, filtering logic, and search functionality implementation.

Requirements:

- Add notes with title and description.
- Tag notes with categories.
- Edit and delete notes.
- Search notes by keyword.
- Filter notes by tags.
- Persist notes using localStorage or Firebase.

Expected Output:
A searchable and categorized knowledge management system with persistent note storage.

PROBLEM STATEMENT 9

Interactive Flashcard Learning System

Problem Statement:
Design a flashcard-based learning application for interactive study sessions.

Objective:
To test animation handling, progress tracking, and structured data storage.

Requirements:

- Add flashcards with question and answer.
- Implement card flip animation.
- Mark flashcards as mastered.
- Track learning progress percentage.
- Persist flashcard data.

Expected Output:
An interactive flashcard system that tracks mastery and stores learning progress.

PROBLEM STATEMENT 10

Digital Certificate Generator

Problem Statement:
Develop a certificate generation system that dynamically creates customized certificates.

Objective:
To evaluate dynamic content rendering, template switching, and file export handling.

Requirements:

- Input participant name, event name, and date.
- Generate certificate preview dynamically.
- Allow selection between multiple certificate templates.
- Provide option to download certificate as image or PDF.
- Persist template preference or last generated data.

Expected Output:
A customizable certificate generator with downloadable output and stored preferences.
