Math game prompt

By DesignersCode

Prompt for Cursor and Polymet: Math Game for Mobile

Objective: Build a simple, beautiful, and modern math mobile game targeting Gen Z users. The app should focus on fun, speed, and an engaging design.

Features and Functionality

1. Core Gameplay

- **Game Modes:**
  - Simple arithmetic operations: addition, subtraction, multiplication.
  - 3 difficulty levels:
    - **Easy:** Operands between 1-10.
    - **Medium:** Operands between 11-50.
    - **Hard:** Operands between 51-100.
- **Questions Timer:**
  - Each question has a countdown timer set for 60 seconds.
  - Show a visible timer bar shrinking or a countdown clock.
- **Answer Input:**
  - Display a virtual keypad with numbers `0-9`, a `"-"` key for negative numbers, and a "Submit" button.
  - User's entered answer is displayed above the number pad in real-time.
  - Clear the input field after each question.
- **Correct Answer Progression:**
  - Allow the user to proceed only when the answer is correct.
  - If the answer is incorrect, freeze progression and show a retry button.

2. Scoring

- Score is based on the number of correct answers given within 60 seconds for a session.
- Keep track of individual session scores and display them at the end.

3. Leaderboard

- Show a leaderboard with the top 10 highest scores globally.
- Allow users to see their ranking.
- Use Firestore for real-time leaderboard updates.

4. User Interface Design

- **Gen Z Aesthetic:**
  - Vibrant gradients, smooth animations, and modern typography.
  - Minimalistic and clean layout with engaging elements (e.g., bouncing buttons, hover effects).
- **Layouts:**
  - Main Screen: Start button and options to select difficulty level.
  - Gameplay Screen:
    - Display the current arithmetic question prominently.
    - Timer bar or countdown clock.
    - Answer input area with a responsive number pad and Submit button.
  - End Screen: Show the user's score and allow sharing.
  - Leaderboard Screen: Display global scores with animated transitions.

Technical Specifications

1. Technology Stack

- **Frontend:** React with modern features (hooks, context API).
- **Backend:** Firestore for the database and leaderboard.

2. Frontend Details

- **Game Logic:** Randomly generate arithmetic questions based on difficulty level.
- **State Management:** Use React hooks to manage user input, score, and timer.
- **Animations:** Leverage CSS animations or libraries like Framer Motion for interactive elements and timer effects.
- **Responsive Design:** Use CSS Flexbox/Grid for responsive layouts across devices.

3. Backend Details

- **Firestore Setup:**
  - Store leaderboard scores in Firestore with fields for `user`, `score`, and `timestamp`.
  - Fetch and update scores in real-time.
- Use Firestore rules to ensure secure updates and retrieval.

Required Code and Design Outputs

1. Frontend Code

- React components for:
    - Main menu with difficulty selection.
    - Question display and input mechanics.
    - Timer countdown with dynamic animations.
    - Score tracking logic.
    - End screen with the score summary.
    - Leaderboard display with top 10 scores and current user score.
- Integrate Firestore SDK to fetch and update leaderboard data.

2. Backend Code

- Firestore schema setup:
    - `leaderboard` collection with documents for user scores.
- Firestore security rules to:
    - Allow authenticated reads for leaderboard data.
    - Restrict writes to prevent manipulation.

3. UI Designs

- High-fidelity mockups or components:
    - Main menu.
    - Gameplay interface with keypad and answer display.
    - Leaderboard screen.
    - End screen.

Example Visual Style

- Use a palette of vibrant gradients like purple-blue or orange-pink.
- Typography: Modern fonts like Poppins or Roboto.
- Buttons: Rounded, soft shadows, hover/press animations.
- Include playful, Gen Z-friendly animations (e.g., bouncing elements, glowing effects).

Additional Requirements

- Code should be modular and easy to maintain.
- Include comments explaining major parts of the logic.
- Incorporate error handling for scenarios like Firestore connectivity issues.