**Prompt for ChatGPT to Build a Flappy Bird Game:**

You are an expert Python developer specializing in Pygame. Write a complete and fully working **Flappy Bird clone** in Python using Pygame. The code should be modular, readable, and error-free. The entire game logic should be included in a **single Python file**, with all required assets and features stubbed or handled gracefully.

**✅ Game Requirements:**

1. **Main Menu**:
   * Include a **visually styled main menu** with options: “Play”, “View High Scores”, “Quit”.
   * Use proper spacing, fonts, and center-aligned options.
2. **Gameplay**:
   * Flappy Bird-style mechanics (tap/spacebar to flap).
   * Gravity pulls the bird down continuously.
   * Pipes (top and bottom) scroll from right to left at a fixed speed.
   * Random vertical gap positions between top and bottom pipes.
   * Bird must pass through pipes without touching them or the ground/ceiling.
   * Display current score during gameplay.
3. **Graphics**:
   * Draw a **bird** (can be a simple colored circle or rectangle if no asset is used).
   * Pipes and background must be clearly visible.
   * Support window resizing using pygame.RESIZABLE.
4. **Pause/Resume**:
   * Press ‘P’ to pause and resume the game.
   * When paused, display a “Paused” message on screen.
5. **Game Over Screen**:
   * When the player hits a pipe or ground, display “Game Over” with options to **Replay** or **Return to Main Menu**.
6. **High Score System**:
   * Store top 5 scores in a local highscores.json file.
   * After game over, if the player’s score is among the top 5, prompt the player to enter their name **in-game** (not via console).
   * Scores should be saved per session and viewable via main menu.
7. **Audio**:
   * Add background music during gameplay.
   * Add flap sound, pipe pass sound, and collision/game over sound.
   * If audio files are not found, handle gracefully (no crash).
8. **Modular Code Structure**:
   * Include the following modular functions:
     + main\_menu(window)
     + game\_loop(window)
     + draw\_bird(...)
     + draw\_pipes(...)
     + check\_collision(...)
     + pause\_game(...)
     + load\_highscores()
     + save\_highscores()
     + get\_player\_name(...)
     + display\_highscores(...)
9. **Asset Handling**:
   * Use rectangles or basic shapes if no image assets are assumed.
   * Define all fonts (title\_font, menu\_font, etc.) properly using pygame.font.Font.
   * Handle highscores.json creation if it doesn’t exist.
10. **Final Requirements**:
    * All required imports (pygame, random, json, os, sys) should be included.
    * No undefined variables or missing functions.
    * Use appropriate frame rate control (pygame.time.Clock()).
    * Include instructions in comments for any asset placement, if needed.

Make sure the full game works after just saving and running the script, assuming Pygame is installed. Handle missing files gracefully with fallbacks or silent fails.