**Project Plan for Building a Poker Game in Python Flask and React/HTML/CSS**

**Week 1: Planning and Initial Development**

**Day 1-2: Requirements and Design**

* **Define Requirements:**
  + Determine game features: different poker variants (Texas Hold'em, Omaha, etc.), multiplayer support, AI opponents, betting system, hand evaluation, etc.
  + Outline UI requirements: table layout, card design, player controls, bet indicators, etc.
* **Design UI Mockups:**
  + Create wireframes/mockups for the poker table and game controls using tools like Figma or Sketch.
  + Define the color scheme and design elements for the game.

**Day 3-4: Set Up Development Environment**

* **Backend Setup (Python Flask):**
  + Set up a Python environment.
  + Install Flask and necessary libraries (e.g., Flask, SQLAlchemy for database management, python-poker for poker logic).
* **Frontend Setup (React/HTML/CSS):**
  + Initialize a React project using Create React App.
  + Set up the folder structure for components, styles, and assets.
  + Install necessary dependencies (e.g., react-router, styled-components).

**Day 5-7: Basic Poker Logic and Table Rendering**

* **Implement Poker Logic in Python:**
  + Create classes for the deck, cards, players, and game logic.
  + Implement hand evaluation, betting rounds, and game state management.
* **Render Poker Table in React:**
  + Create the poker table component in React.
  + Render the initial table layout with placeholder cards and player positions using HTML and CSS.
  + Style the table and cards according to your design.

**Week 2: Core Functionality and Integration**

**Day 8-10: Game State Management and Interaction**

* **Implement Game State in React:**
  + Set up state management (using useState or redux) to handle table state, player states, and game status.
  + Implement click/touch events for betting, folding, and other player actions.
* **Integrate Backend and Frontend:**
  + Set up API endpoints in Flask for game state, player actions, and game progress.
  + Fetch and update the game state from the frontend using fetch or axios.

**Day 11-13: Additional Features**

* **Add Game Rules:**
  + Implement rules for different poker variants (e.g., Texas Hold'em).
  + Display game status on the frontend (e.g., current player, pot size, player actions).
* **Multiplayer Mode:**
  + Allow multiple players to join the same game session.
  + Implement game controls (e.g., start game, reset game, add/remove players).

**Day 14: Styling and Responsiveness**

* **Enhance UI/UX:**
  + Improve the styling of the poker table and cards.
  + Ensure the UI is responsive and works well on different devices.
  + Add animations for card dealing and player actions.

**Week 3: Testing, Optimization, and Deployment**

**Day 15-17: Testing and Bug Fixing**

* **Unit Testing:**
  + Write unit tests for poker logic in Python.
  + Test API endpoints to ensure they return correct responses.
* **Frontend Testing:**
  + Test the React components and interactions.
  + Fix any bugs or issues found during testing.

**Day 18-19: Optimization**

* **Optimize Backend:**
  + Improve performance of poker logic and API endpoints.
* **Optimize Frontend:**
  + Optimize rendering and state management in React.
  + Ensure smooth animations and interactions.

**Day 20-21: Deployment**

* **Prepare for Deployment:**
  + Set up a production environment for the backend (e.g., deploy to Heroku, AWS).
  + Build and deploy the React frontend (e.g., deploy to Netlify, Vercel).
* **Final Testing:**
  + Perform final testing in the production environment.
  + Ensure all features work as expected and the game is fully functional.

**Deliverables:**

* Fully functional poker game with basic features.
* Responsive and styled user interface.
* Deployed and accessible poker game.

**Tools and Technologies:**

* **Backend:** Python, Flask, SQLAlchemy
* **Frontend:** React, HTML, CSS, styled-components
* **Testing:** PyTest for backend, Jest for frontend
* **Deployment:** Heroku/AWS for backend, Netlify/Vercel for frontend