1. Introduction

1.1 Project Overview

This document outlines the business and functional requirements for a real-time, online cards game named **Fruit Cards**. The game is designed for 3 to 7 players and features a unique "pass the card" mechanic where the goal is to be the first to collect all cards of a single, specified fruit type. The game will be accessible via a shareable room code or link, requiring no user authentication.

1.2 Project Goals

- **Create an Engaging Experience:** Develop a simple, intuitive, and fun card game that is easy for new players to learn and play.
- **Enable Real-time Multiplayer:** Use a WebSocket-based architecture to provide a seamless, low-latency, real-time experience for all players in a game room.
- **Ensure Accessibility:** The game must be accessible without a complex sign-up process, relying on room codes for quick entry.
- **Establish a Foundational System:** Build a robust and scalable technical foundation that can support multiple concurrent games.

2. Scope

2.1 In Scope

- **Game Lobby:** Creation of a new game room, sharing a room code or link, and joining an existing room.
- **Player Management:** Support for 3 to 7 players per game, with automatic or user-defined nicknames.
- **Game Setup:** A "leader" player can configure the game's deck by selecting the card types to be used.
- **Core Gameplay Loop:** Implementation of card passing, turn management, and win condition detection.
- **Real-time Communication:** All game state changes will be communicated in real-time to all connected players.
- **Basic UI:** A clean, responsive user interface displaying the game state, player hands, and turn information.
- **Game End:** Detection of a winner and an end-of-game screen displaying the results.
- **Persistent Sessions:** Players will be able to reconnect to their ongoing game if they refresh the page or disconnect briefly.

2.2 Out of Scope

- Authentication or user accounts (e.g., email/password login).
- Chat functionality within the game room.
- Persistent user data (e.g., win/loss history, player profiles).
- In-game currency or microtransactions.
- Advanced AI for single-player modes.
- Spectator mode.

3. Functional Requirements

3.1 Game Lobby Requirements

- **FR-01:** The system shall allow a user to create a new game room and become the "leader."
- **FR-02:** Upon creation, the system shall generate a unique, short, and easily shareable room code (e.g., a 4-digit number or a short alphanumeric string).
- **FR-03:** The leader shall be able to share a direct link to the game room that includes the room code.
- **FR-04:** Users shall be able to join a game room by entering the correct room code on a landing page.
- **FR-05:** Players must provide a nickname upon joining. If a nickname is not provided, one shall be auto-generated (e.g., "Player_123").
- **FR-06:** The lobby shall display a list of all connected players' nicknames.
- **FR-07:** The game leader shall have the exclusive ability to start the game.

3.2 Game Setup Requirements

- **FR-08:** The game leader shall be able to choose between 3 and 5 unique card types for the game deck.
- **FR-09:** The number of players in the game must be between 3 and 7, inclusive.

3.3 Core Gameplay Requirements

- **FR-10:** The backend shall create a game deck with a total number of cards equal to (number of players) * (number of cards per type) + 1.
- **FR-11:** Each player shall be dealt an equal number of cards at the start of the game, n, and the first player to take their turn shall receive n+1 cards.
- **FR-12:** The UI shall clearly indicate which player's turn it is.
- **FR-13:** When it is a player's turn, the UI shall allow them to select one card from their hand to "pass."
- **FR-14:** A "pass" button shall become active after a card is selected.
- **FR-15:** When a card is passed, the system shall add the passed card to the hand of the

next player in the turn order.

- **FR-16:** The turn shall automatically proceed to the next player after a card is passed.
- **FR-17:** The game shall end immediately when a player collects all cards of a single type.
- **FR-18:** The system shall declare the winning player and display a winning message to all players.
- **FR-19:** The UI shall display a countdown timer or loader for the current player's turn.
- **FR-20:** The duration of the timer shall be defined by the game leader during setup (or a default value).
- **FR-21:** If the timer runs out, the system shall automatically select and pass a random card from the current player's hand to the next player.
- **FR-22:** The UI shall provide a visual indication of the timer countdown.

3.4 Session Management

- **FR-23:** The system shall create a persistent session for each player, allowing them to reconnect to their game room after a refresh or brief disconnection.
- **FR-24:** The system shall use a unique, client-side identifier (like a cookie or local storage key) to associate a player with their active session.
- **FR-25:** Upon reconnecting, the player shall automatically rejoin the game room and their previous state (hand, turn status, etc.) shall be restored.

3.5 Audio Requirements

- **FR-26:** The game shall play distinct sound effects for key events, such as:
  - A card being passed.
  - A turn starting.
  - A player joining the room.
  - A player winning the game.

3.6 User Interface and Experience Requirements

This section defines the visual design and navigational flow for the game.

- **Page Navigation & User Flow:**
  - **Landing Page:** The user arrives here and can choose to either create a new game or join an existing one.
  - **Lobby Page:** Players navigate here after creating or joining a game room.
  - **Game Page:** All players are automatically navigated here when the game starts.
  - **Game Over Page:** The game ends for all players when a winner is declared, and they are navigated to a final results screen.
  - **Restart/Return:** From the Game Over page, players can choose to start a new game or return to the landing page.
- **Landing Page (Home):**
  - **Layout:** Centered content with a clean, minimalist design that is responsive for both mobile and desktop.
  - **Title:** "Fruit Cards" prominently displayed.

    - **Main Actions:** Two distinct, touch-friendly buttons: "Create Game" and "Join Game."
    - **Input Field:** A clear, rounded text input field appears when "Join Game" is clicked.
- **Lobby Page:**
    - **Layout:** A central container with a backdrop of a subtle, blurred image of fruits.
    - **Room Code:** The unique room code is displayed at the top.
    - **Player List:** A list of connected players, each with their nickname. The leader's nickname will have a "Leader" tag.
    - **Game Settings (Leader Only):** The leader sees options to choose fruit types and a "Start Game" button.
    - **Player View (Non-Leader):** Other players will see a message indicating they are waiting for the leader.
- **Game Page:**
    - **Layout:** A semi-circular layout for the other players' hands, with the current player's hand at the bottom of the screen.
    - **Player's Hand:** The player's hand is laid out horizontally, and cards are selectable.
    - **Other Players' Hands:** Represented by a smaller stack of face-down cards and their nickname. The current player's turn is highlighted.
    - **Turn Timer:** A visual loader or progress bar displays the countdown.
    - **Action Button:** A "Pass Card" button is available when a card is selected.
    - **Game Messages:** A text area or modal announces key events.
- **Game Over Page:**
    - **Layout:** A celebratory design with a prominent winner announcement.
    - **Winner Announcement:** The winner's nickname is displayed with a message like "is the winner!"
    - **Final Hands:** Optionally, all players' final hands can be displayed.
    - **Actions:** "Play Again" and "Return to Home" buttons are available.

3.7 Technical Requirements

- **TR-01:** The frontend shall be developed using **React**, **Vite**, **Tailwind CSS**, and **React Router**.
- **TR-02:** The backend shall be developed using **Node.js** and **Express**.
- **TR-03:** Real-time communication between the frontend and backend shall be managed using **Socket.IO**.
- **TR-04:** Game room state and player information shall be stored persistently using **SQLite**.

3.8 Navigation Requirements

- **FR-27:** The frontend shall use a routing library, such as React Router, to manage navigation between the game's various pages (Landing, Lobby, Game, and Game Over).

4. Non-Functional Requirements

- **NFR-01 (Performance):** The latency for real-time card passing and turn updates shall be minimal (less than 500ms).
- **NFR-02 (Scalability):** The backend architecture shall be designed to handle a moderate number of concurrent games (e.g., 50-100 games at a time) without significant performance degradation.
- **NFR-03 (Usability):** The user interface shall be simple, intuitive, and require no prior knowledge of the game to get started.
- **NFR-04 (Security):** The backend must prevent players from manipulating game state (e.g., cheating by changing their hand) and ensure only authorized players can join a room.
- **NFR-05 (Responsiveness):** The user interface shall be fully responsive and optimized for both desktop and mobile devices.
- **NFR-06 (Localization):** The application must support English and Arabic for all user-facing text and UI elements.
- **NFR-07 (Timing):** The timer implementation must be synchronized across all clients to ensure consistent turn-time enforcement.
- **NFR-08 (Audio Quality):** Audio effects shall be clear, non-intrusive, and load quickly to enhance the user experience.
- **NFR-09 (Reliability):** The game shall be resilient to client-side disconnections and refreshes, maintaining the player's session and re-establishing their connection automatically.

5. Data Model

5.1 Entities and Relationships

- **Game Room:**
  - roomId (Primary Key, unique room code)
  - leaderId (ID of the game leader)
  - gameStatus (e.g., 'lobby', 'in-progress', 'completed')
  - cardTypes (JSON array of fruit types chosen by the leader)
  - timerDuration (Integer, countdown time for each turn)
- **Player:**
  - playerId (Unique ID for the socket connection)
  - sessionId (Unique, persistent ID stored client-side for session recovery)
  - nickname (Player's chosen or auto-generated name)
  - roomId (Foreign Key to Game Room)
  - hand (JSON array of cards in the player's hand)
- **Card:** (Implicitly handled within the Player's hand property)

- cardType (e.g., "Apple", "Banana", "Cherry")
- cardId (Unique identifier for the specific card)