High- and Low-Level Design

Texas Hold‘em Application

Contents

[Section 1: Hardware & Architecture 2](#_Toc150108138)

[Section 2: Security 2](#_Toc150108139)

[Section 3: Database Design 3](#_Toc150108140)

[Section 4: Data Flow 5](#_Toc150108141)

[Section 5: Class Diagram 6](#_Toc150108142)

[Section 6: User Interface 7](#_Toc150108143)

[Section 7: Activity Diagram 9](#_Toc150108144)

# Section 1: Hardware & Architecture

**Hardware**

For the server-side hardware, we will be using Microsoft Azure App service, with a database on Azure Databases for MySQL. This server will be built using C# and ASP.NET Core. For the client-side hardware, the application will be running on any user devices that are able to run React Native applications.

**Architecture**

The architectures involved with building this program are a combination of Client/Server Architecture and Component-Based Architecture. The client/server architecture aspect is due to the separation between the client-side frontend of React Native and the server-side backend of C# and ASP.NET Core. The Component-Based architecture aspect is due to the program being a collection of loosely based components providing services to each other, like game logic, database communication, user authentication, etc.

# Section 2: Security

**User Login/Registration:**

1. **Account Creation:**
   * **Invite Code Verification:** Require users to enter a valid invite code to create an account. This helps control the number of accounts and prevents unauthorized access.
   * **Unique Username and Email:** Enforce unique usernames and email addresses for each account. This ensures user identity and prevents account duplication.
   * **Strong Password Requirements:** Implement strong password requirements, including minimum length, complexity, and password history. This protects user accounts from brute-force attacks and weak passwords.
2. **User Authentication:**
   * **Secure Login:** Implement secure login mechanisms, such as salted and hashed passwords, to protect user credentials.
   * **Multi-Factor Authentication (MFA):** Offer MFA options, such as email verification or SMS codes, to add an extra layer of security.
   * **Login Throttling:** Implement login throttling limits to prevent brute-force attacks.
3. **Session Management:**
   * **Session Timeouts:** Set appropriate session timeouts to automatically log out inactive users.
   * **Secure Session Tokens:** Use secure session tokens and invalidate tokens upon logout or session expiration.
   * **Prevent Session Hijacking:** Implement measures to prevent session hijacking, such as session cookies with secure flags and HTTP Strict Transport Security (HSTS).

**Password Resets:**

1. **Password Reset Requests:**
   * **Verify User Identity:** Verify user identity before initiating a password reset. This can be done by sending a verification code to the user's registered email address.
   * **Time-Sensitive Reset Links:** Generate time-sensitive reset links to prevent unauthorized password changes.
   * **Prevent Password Reuse:** Discourage password reuse by requiring users to create a new password that is different from their previous passwords.
2. **Password Storage:**
   * **Never Store Plaintext Passwords:** Never store passwords in plaintext. Always store passwords in a securely hashed format.
   * **Use Strong Hashing Algorithms:** Use strong hashing algorithms, such as bcrypt or Argon2, to protect password hashes.
   * **Regular Password Hashing Updates:** Consider regularly updating password hashes with newer, more secure hashing algorithms.

**Invite Code System:**

1. **Invite Code Generation:**
   * **Unique Invite Codes:** Generate unique and unpredictable invite codes to prevent unauthorized account creation.
   * **Invite Code Expiration:** Set expiration dates for invite codes to limit their usage and prevent code sharing.
   * **Invite Code Revocation:** Allow administrators to revoke invite codes if they are compromised or no longer needed.
2. **Invite Code Usage:**
   * **Invite Code Validation:** Validate invite codes before allowing account creation.
   * **Invite Code Consumption:** Mark invite codes as used after successful account creation to prevent multiple accounts per code.
   * **Invite Code Tracking:** Keep track of invite code usage to monitor account creation and identify potential misuse.

# Section 3: Database Design

**Database Name:** GamblersAnonymous

**Purpose:** To store and manage data for a poker game application.

**Normalization Level:** Third Normal Form (3NF)

**Tables:**

1. **Players Table:** Primary Key: playerID Columns:
   * playerName: VARCHAR(255)
   * playerBlacklist: BOOLEAN
   * avatar: VARCHAR(255)
   * virtualCurrencyBalance: DECIMAL(10,2)
   * playerStatus: ENUM('active', 'folded', 'waiting')
   * playerRank: INTEGER
2. **Games Table:** Primary Key: gameID Columns:
   * roundNumber: INTEGER
   * limitType: ENUM('limit', 'no-limit', 'pot-limit')
   * smallBlind: DECIMAL(10,2)
   * bigBlind: DECIMAL(10,2)
   * currentPot: DECIMAL(10,2)
   * dealerPosition: INTEGER
   * playerID: INTEGER (Foreign key referencing Players table)
3. **CommunityCards Table:** Primary Key: communityCardsID Columns:
   * gameID: INTEGER (Foreign key referencing Games table)
   * card1: VARCHAR(255)
   * card2: VARCHAR(255)
   * card3: VARCHAR(255)
   * card4: VARCHAR(255)
   * card5: VARCHAR(255)
4. **PlayerCards Table:** Primary Key: playerCardsID Columns:
   * gameID: INTEGER (Foreign key referencing Games table)
   * playerID: INTEGER (Foreign key referencing Players table)
   * card1: VARCHAR(255)
   * card2: VARCHAR(255)
5. **BettingActions Table:** Primary Key: bettingActionsID Columns:
   * gameID: INTEGER (Foreign key referencing Games table)
   * roundNumber: INTEGER
   * playerID: INTEGER (Foreign key referencing Players table)
   * actionType: ENUM('bet', 'raise', 'fold', 'check')
   * actionAmount: DECIMAL(10,2)
6. **GameResults Table:** Primary Key: gameResultsID Columns:
   * gameID: INTEGER (Foreign key referencing Games table)
   * winningPlayerID: INTEGER (Foreign key referencing Players table)
   * winningHand: VARCHAR(255)
   * potAmountWon: DECIMAL(10,2)

**Relationships:**

1. One-to-Many: Players table to Games table (One player can participate in many games)
2. One-to-Many: Games table to CommunityCards table (One game has one set of community cards)
3. Many-to-Many: Games table to Players table (Many players can participate in one game)
4. Many-to-Many: Games table to BettingActions table (Many betting actions can occur in one game)
5. One-to-One: Games table to GameResults table (One game has one game result)

**Additional Considerations:**

1. Implement appropriate data types for each column to ensure data integrity.
2. Implement appropriate indexes for frequently queried columns to improve performance.
3. Consider implementing triggers to maintain data consistency across tables.
4. Implement appropriate security measures to protect sensitive data.

This database design should provide a solid foundation for managing data for a poker game application. It adheres to 3NF principles, ensuring data integrity and minimizing redundancy. The design can be further extended to accommodate additional features and requirements as needed.

# Section 4: Data Flow

A diagram of a game

Description automatically generated

# Section 5: Class Diagram

A screenshot of a computer screen

Description automatically generated

# Section 6: User Interface

**Overview:**

This document outlines the High-Level Design information for the Texas Holdem app's User Interface component. We are aiming to create a user-centric, engaging, and seamless gaming experience that focuses on specifying the 'what' aspects of the design, detailing design choices and justifying architectural decisions based on the user interface requirements.

**Architecture:**

The user interface for the Texas Holdem app will be built on a 3-tier architecture, which includes the Presentation, Logic, and Data tiers. This separation ensures that the user interface (Presentation layer) can evolve independently from the other two, allowing for greater scalability and flexibility. This architecture improves maintenance and future growth, critical for the app’s long-term success.

Reference for 3-tier architecture: IBM Three-Tier Architecture  
 <https://www.ibm.com/cloud/learn/three-tier-architecture>

**Security:**

Security measures for the web-app include secure login procedures for third-party authentication, ensuring that sensitive user data is protected. Logging of sensitive operations is limited to respect user privacy, critical actions within the app are audited to ensure accountability and to enhance security.

An IP-based firewall rule is in place to secure access to the web application, only permitting traffic from trusted sources.

**System Entities:**

The class diagram on page 7 identifies system nouns and outlines preliminary relationships and dependencies.

**Hardware:**

The Texas Holdem app is designed to for web access; the app will be compatible with all modern browsers.

**External Systems:**

The app will interface with a third-party authentication provider to enable logins through services like GitHub and Google.

**User Interface:**

The navigation is structured to be intuitive and responsive, with a focus on a seamless transition from lobby to game rooms. Mock-ups from design sessions are attached to provide a conceptual visualization of the interface.

A screenshot of a game

Description automatically generated

**Data Storage:**

Data will be handled using the XYUL database system, a relational database known for its robustness and scalability. A high-level conceptual model of the database design is included, highlighting the tables and their primary relationships.

**Other Outputs**:

Email notifications for user-related events, such as registration confirmations and password resets.

Reports:

**Potential report features for system analytics include:**

Most popular and least popular pages.

User engagement metrics based on browser and hardware type.

Daily and monthly active user reports.

This High-Level Design document serves as a foundation for the detailed design phase, where each aspect will be fleshed out with precise specifications and implementation details.

# Section 7: Activity Diagram

A diagram of a game

Description automatically generated

A diagram of a game

Description automatically generated