Blackjack Bounty

A Revolutionary Blockchain Game

Executive Summary

Blackjack Bounty is a revolutionary mobile Blackjack game that combines classic card gaming with blockchain transparency and rewards. Built with Unity for a great mobile experience, a Node.js backend for seamless gameplay, and the Sui blockchain for a fair game and rewards, it targets casual gamers and crypto enthusiasts. Our currency and play system features in-game currency for casual play, IAP Tickets for premium play, and $MOLA tokens for blockchain rewards. This provides a true solution for a blockchain game economy, and a great accessibility. We aim to capture a share of the $100B mobile gaming market and the $10B blockchain gaming sector.

Funding Goal: $500,000 to complete development, launch marketing, and ensure compliance.

Contacts: mola@molaict.com – www.molaict.com

Contents

1 Project Overview

Blackjack Bounty is a mobile Blackjack game designed for fun, fairness, and real world rewards. Key features include:

- **Unity Client**: Polished UI and local game logic for iOS and Android.
- **Sui Blockchain**: Transparent card randomization and $MOLA rewards via smart contracts.
- **Backend Layer**: Manages game state, currencies, and blockchain interactions.
- **Currency and Play System**:
  - **Game Gold**: For casual rooms and events, purchasable via IAP or earned in-game.
  - **Tickets**: For premium rooms and events, purchasable via IAP.
  - **$MOLA**: Earned via events and premium rooms, convertible to $SUI, non-purchasable/tradable token.
- **Play Sections**:
  - **Casual Rooms**: Non-blockchain gameplay with Game Gold.
  - **Casual Events**: Competitive play with Game Gold, lower $MOLA bounties.
  - **Premium Rooms**: Blockchain-tracked gameplay with Tickets, $MOLA bounties.
  - **Premium Events**: Competitive play with Tickets, higher $MOLA bounties.
- **Regulatory Compliance**: $MOLA to $SUI conversions framed as loyalty rewards.

2 Problem Statement

The mobile gaming industry, valued at over $100B, faces challenges:

2.1 Traditional Gaming Problems

- **Untrustable Randomization**: Hidden card distribution processes destroys player trust.
- **Regulatory Hurdles**: Real-money rewards risk gambling classification.
- **Limited Rewards**: Lack of real-money rewards without gambling classification restricts engagement.
- **High Cashout Barriers**: Excessive thresholds and fees limit player comfort.
- **Centralized Storage**: Reliance on centralized servers increases costs, risks data breaches, and enables censorship.

2.2 Blockchain Gaming Problems

- **Complexity**: Blockchain games alienate casual players with complex mechanics.
- **Mobile Platform Policies**: Mobile app platforms do not support direct token transactions or external wallet integrations.
- **Delayed & Unengaging Rewards**: Blockchain games often delay rewards until major achievements or distant airdrops, reducing engagement.
- **Low Quality Gameplay**: Fails to catch the quality of mainstream mobile games.

3 Solution: Blackjack Bounty

Blackjack Bounty redefines mobile gaming with:

- **Engaging Gameplay**: Classic Blackjack with great gameplay, built in Unity.
- **Transparent Randomization**: Sui's smart contracts can use Verifiable Random Functions (VRF), publicly verifiable. You can trust the casino now!
- **Play Sections**:
  - **Casual Rooms**: Game Gold-based, non-blockchain play.
  - **Casual Events**: Game Gold-based, rewarding $MOLA (lower amounts).
  - **Premium Rooms**: Ticket-based, blockchain-tracked, $MOLA bounties.
  - **Premium Events**: Ticket-based, higher $MOLA bounties.
- **Backend Sync**: Node.js backend manages game state and currencies.

- **Regulatory Safety**: $MOLA to $SUI conversions limited and framed as loyalty bounties.
- **Seamless Authentication**: Sui zkLogin enables OAuth-based logins (e.g., Google, Facebook).
- **Decentralized Storage**: Walrus stores game data on-chain for censorship resistance and low costs.

This ensures accessibility for casual players and rewards for crypto users, maintaining fairness and compliance.

4 Currency and Play System

Blackjack Bounty uses three currencies and four play sections for diverse audiences:

4.1 Game Gold (Casual Currency)

- **Purpose**: For casual rooms and events (e.g., leaderboards, tournaments).
- **Acquisition**:
  - Purchasable via IAP.
  - Earnable in-game (e.g., daily logins, achievements).
- **Characteristics**: Non-blockchain, non-convertible, affordable.
- **Casual Events**: Weekly/daily leaderboards and tournaments with Game Gold entry fees. Top 20% earn $MOLA tokens.

4.2 Tickets (Premium Currency)

- **Purpose**: For Premium Rooms and Premium Events (e.g., leaderboards, tournaments).
- **Acquisition**: Purchasable via IAP.
- **Characteristics**: Non-blockchain, non-convertible, premium access.
- **Premium Events**: Weekly/daily leaderboards and tournaments with Ticket entry fees. Top 10% earn higher $MOLA tokens.

4.3 $MOLA (Bounty tokens)

- **Purpose**: For conversion to $SUI, earned via casual events (lower rewards) and premium rooms/events (higher rewards).
- **Acquisition**: Earned through:
  - Casual Events (e.g., 1–5 $MOLA for top 10%).
  - Premium Rooms (e.g., 1–3 $MOLA per win).

  - Premium Events (e.g., 5–20 $MOLA for top 10%).

- **Characteristics**: Blockchain-based, non-purchasable, non-tradable.
- **Conversion**: Convertible to $SUI with a low threshold.

4.4 Play Sections

- **Casual Rooms**: Game Gold-based, non-blockchain gameplay, no $MOLA rewards.
- **Casual Events**:
  - Weekly Leaderboard
  - Daily Leaderboard
  - Global Tournaments
  - Mini Tournaments (5 players)
- **Premium Rooms**: Ticket-based, blockchain-tracked.
- **Premium Events**:
  - Weekly Leaderboard
  - Daily Leaderboard
  - Global Tournaments
  - Mini Tournaments (5 players)

This system ensures game itself is not tricking players and they enjoy affordable play with in-game gold, earning real world bounties all within regulatory boundaries.

5 Avoiding Gambling Classification

Blackjack Bounty avoids gambling classification through:

- **Casual Play**: Game Gold and Tickets fund Casual Rooms/Events, purely for entertainment.
- **Premium Play**: Tickets fund Premium Rooms/Events, with Mola token rewards named as bounty.
- **Conversion Limits**:
  - $MOLA tokens have a low minimum threshold for conversion to $SUI, ensuring accessibility for players.
- **Legal Framing**: Consulted experts for regional compliance.
- **Transparency**: Sui's public contracts ensure verifiable Premium outcomes.

This positions Blackjack Bounty as a skill-based, rewarding game, not gambling.

6 Technical Architecture

Blackjack Bounty's modular architecture comprises four layers:

6.1 Unity Client

- **Role**: Handles UI, local logic, play sections, and authentication.
- **Components**:
  - UI rendering (cards, scores, balances).
  - Local Blackjack rules for casual rooms.
  - REST API calls for game state, currencies, and events.
  - Sui zkLogin for OAuth-based authentication (e.g., Google, Facebook).

6.2 Backend (Node.js)

- **Role**: Manages game state, currencies (Game Gold, Tickets, $MOLA tokens), events, Sui interactions, and storage.
- **Components**:
  - REST API for Unity communication.
  - Redis for caching balances and states.
  - PostgreSQL for off-chain or game centric temporary data storage.
  - Sui SDK for blockchain operations and zkLogin.
  - Walrus SDK for decentralized storage.

6.3 Sui Blockchain

- **Role**: Runs smart contracts for $MOLA token rewards, $SUI conversions, and authentication.
- **Components**:
  - Smart contract for randomization, $MOLA minting, burning and conversions.
  - Verifiable Random Function (VRF) for shuffling.
  - Sui zkLogin for authentication.
  - Public game data post-game.

6.4 Walrus Storage

- **Role**: Decentralized storage for game data.
- **Components**:
  - Stores profiles, event states, leaderboards and game data.
  - Integrates via Walrus SDK.
  - Ensures availability and low costs.

6.5 Data Flow

1. Unity requests authentication (zkLogin) or actions (e.g., join event) via backend APIs.
2. Backend processes authentication or actions, fetching cards from Sui for Premium play.
3. Sui contract randomizes cards, records outcomes; Walrus stores data.
4. Backend syncs results to Unity, updating balances in Redis/PostgreSQL and Walrus.

7 Implementation Details

7.1 Unity Client (C# Scripts)

1. **GameManager.cs** (Controller)
   - Manages game state, play sections.
   - Methods: `StartGame`, `JoinEvent`, `SyncState`.
2. **BlackjackLogic.cs** (Model)
   - Blackjack rules.
   - Methods: `DealCards`, `Hit`, `Stand`, `GetScore`.
3. **UIManager.cs** (View)
   - Updates UI (cards, balances, results).
   - Methods: `DisplayCards`, `UpdateBalance`, `ShowResult`.
4. **CurrencyManager.cs** (Model)
   - Tracks Game Gold, Tickets, Mola tokens.
   - Methods: `SpendGameGold`, `SpendTickets`, `RequestConversion`.
5. **NetworkManager.cs** (Controller)
   - API communication.
   - Methods: `GetAsync`, `PostAsync`.
6. **Card.cs** (Model)
   - Card with `suit: string`, `value: int`.
7. **AuthManager.cs** (Controller)
   - Sui zkLogin authentication.
   - Methods: `LoginWithOAuth`, `GetZkLogintoken`, `SyncAuthState`.

7.2 Backend (Node.js with Express)

7.2.1 REST API Endpoints

1. **GET /player/:id/balance**
   - Returns balances.
   - Response: `{ gameGold: 1000, tickets: 10, molatokens: 50 }`.
2. **POST /game/start**
   - Starts game in room.
   - Request: `{ playerId: "123", section: "casualRoom" | "premiumRoom" wager: 10, currency: "gameGold" | "ticket" }`.
   - Response: `{ gameId: "abc123", initialHand: { player: [...], dealer: [...] } }`.
3. **POST /game/action**
   - Processes actions.
   - Request: `{ gameId: "abc123", action: "hit" | "stand" }`.
   - Response: `{ updatedHand: [...], status: "ongoing" | "finished", result: "win" | "loss" | "draw" }`.
4. **POST /game/end**
   - Finalizes game, awards Mola tokens for Premium Room wins.
   - Request: `{ gameId: "abc123" }`.
   - Response: `{ result: "win", reward: 2 }`.
5. **POST /event/join**
   - Joins Event.
   - Request: `{ playerId: "123", section: "casualEvent" | "premiumEven entryFee: 100, currency: "gameGold" | "ticket" }`.
   - Response: `{ eventId: "xyz123", status: "joined" }`.
6. **POST /currency/purchase**
   - Handles IAP.
   - Request: `{ playerId: "123", currency: "gameGold" | "ticket", amount: 100, iapReceipt: "xyz" }`.
   - Response: `{ success: true, newBalance: { gameGold: 1100, tickets: 10, molatokens: 50 } }`.
7. **POST /currency/convert**
   - Converts $MOLA tokens to SUI.

    - Request: `{ playerId: "123", amount: 10 }`.
    - Response: `{ success: true, suiAmount: 2 }`.

8. **POST /auth/zklogin**
    - Authenticates via zkLogin.
    - Request: `{ provider: "google" | "facebook", token: "xyz" }`.
    - Response: `{ success: true, playerId: "123", zktoken: "abc" }`.

9. **POST /storage/save**
    - Saves data to Walrus.
    - Request: `{ playerId: "123", data: { gameState: {...} } }`.
    - Response: `{ success: true, storageId: "xyz123" }`.

10. **GET /storage/retrieve/:id**
    - Retrieves data from Walrus.
    - Response: `{ success: true, data: { gameState: {...} } }`.

7.2.2 Services

1. **playerService.js**: `getBalance`, `updateBalance`.
2. **gameService.js**: `startGame`, `processAction`, `endGame`.
3. **eventService.js**: `joinEvent`, `endEvent`, `awardRewards`.
4. **currencyService.js**: `purchaseCurrency`, `convertToSui`.
5. **suiClient.js**: `getRandomCards`, `mintMolatokens`, `convertMolatokens`.
6. **authService.js**: `authenticateZkLogin`, `refreshZktoken`.
7. **storageService.js**: `saveToWalrus`, `retrieveFromWalrus`.

7.3 Sui Blockchain (Move Language)

7.3.1 Smart Contract: BlackjackContract.move

1. **Structs**:
    - `Player { id: UID, molatokens: u64 }`: Tracks Mola tokens.
    - `Game { id: UID, playerId: UID, wager: u64, cards: vector<Card>` `status: u8 }`: Game state.
    - `Card { suit: u8, value: u8 }`: Card representation.

2. **Functions**:

- `init_game(player: &signer, wager: u64)`: Starts Premium Room game, shuffles with VRF.
- `deal_card(gameId: UID)`: Returns card.
- `end_game(gameId: UID, result: u8)`: Awards Mola tokens for wins.
- `mint_mola_tokens(player: &signer, amount: u64)`: Mints $MOLA for events/rooms.
- `convert_to_sui(player: &signer, amount: u64)`: Converts $MOLA tokens.
- `authenticate_zklogin(oauthtoken: vector<u8>)`: Maps OAuth to Sui address.

3. **Events**:

   - `GameStarted { gameId, playerId, wager }`.
   - `GameEnded { gameId, result, reward }`.
   - `Conversion { playerId, amount, suiAmount }`.

8 Market Opportunity

- **Mobile Gaming**: $100B+, 10% YoY growth.
- **Blockchain Gaming**: Exceeded $4.6 billion in 2022 and is poised to hit around $65.7 billion by the end of 2027.
- **Blackjack Appeal**: Universal, suits casual and crypto players.

9 Revenue Model

Revenue streams include:

- **In-App Purchases**:
  - Game Gold (IAP)
  - Tickets (IAP)
- **Projected Revenue**:
  - Year 1: $2M
  - Year 2: $5M

10 Roadmap

- **Q3 2025**: Game prototype, complete backend.
- **Q4 2025**: Beta & testnet.
- **Late Q4 2025**: Global iOS/Android launch, mainnet.

11 Team

- Alper Tuğşad Meydan – Game & Blockchain Developer
- Öncül Maral – Software Engineer
- Assoc. Prof. Adnan Özsoy – Advisor

12 Funding Ask

- **Amount**: $500,000.
- **Use of Funds**:
  - **50% Development**
  - **30% Marketing**: User acquisition for casual/crypto audiences.
  - **20% Legal & Operations**: Compliance, infrastructure costs.

13 Conclusion

Blackjack Bounty blends classic Blackjack with blockchain innovation. Our game economy model is a sustainable solution for mobile blockchain gaming smart contracts players can trust the game itself. Scalable architecture, clear revenue model, and passionate and experienced team, we're set to capture the mobile and blockchain gaming markets. Join us to bring Blackjack Bounty to millions worldwide.