# SolScore: FPL-Based Web3 Prediction Platform

## Requirements

On this first version of the app;

- The app will allow a user to import his/her Wallet
- The app will allow a user to import his/her FPL ID
- The app will allow a user to place stakes on real-time football events
- The app will allow a user to un-stake his earnings within certain intervals
- The app will deduct certain charges and penalties if user withdraws out of the stipulated period.
- The app will monitor real-time events based on users' FPL ID
- The app will reward the user with rewards based on real-time event outcomes

## Core Programs

### 1. FPL Manager Program
- Handles FPL ID verification and connection
- Manages team data synchronisation
- Processes weekly score updates
- Interfaces with external FPL API

### 2. Stake Program
- Manages stake creation and positions
- Handles SOL deposits and withdrawals
- Creates and manages stake PDAs
- Enforces lock periods and penalties
- Tracks stake durations and amounts

### 3. Reward Program
- Calculates performance-based rewards
- Manages reward distribution
- Handles reward pool management
- Processes reward claims

**4. System Treasury Program**
- Manages protocol fees
- Handles vault operations
- Controls protocol reserves
- Manages protocol revenue

# Overview

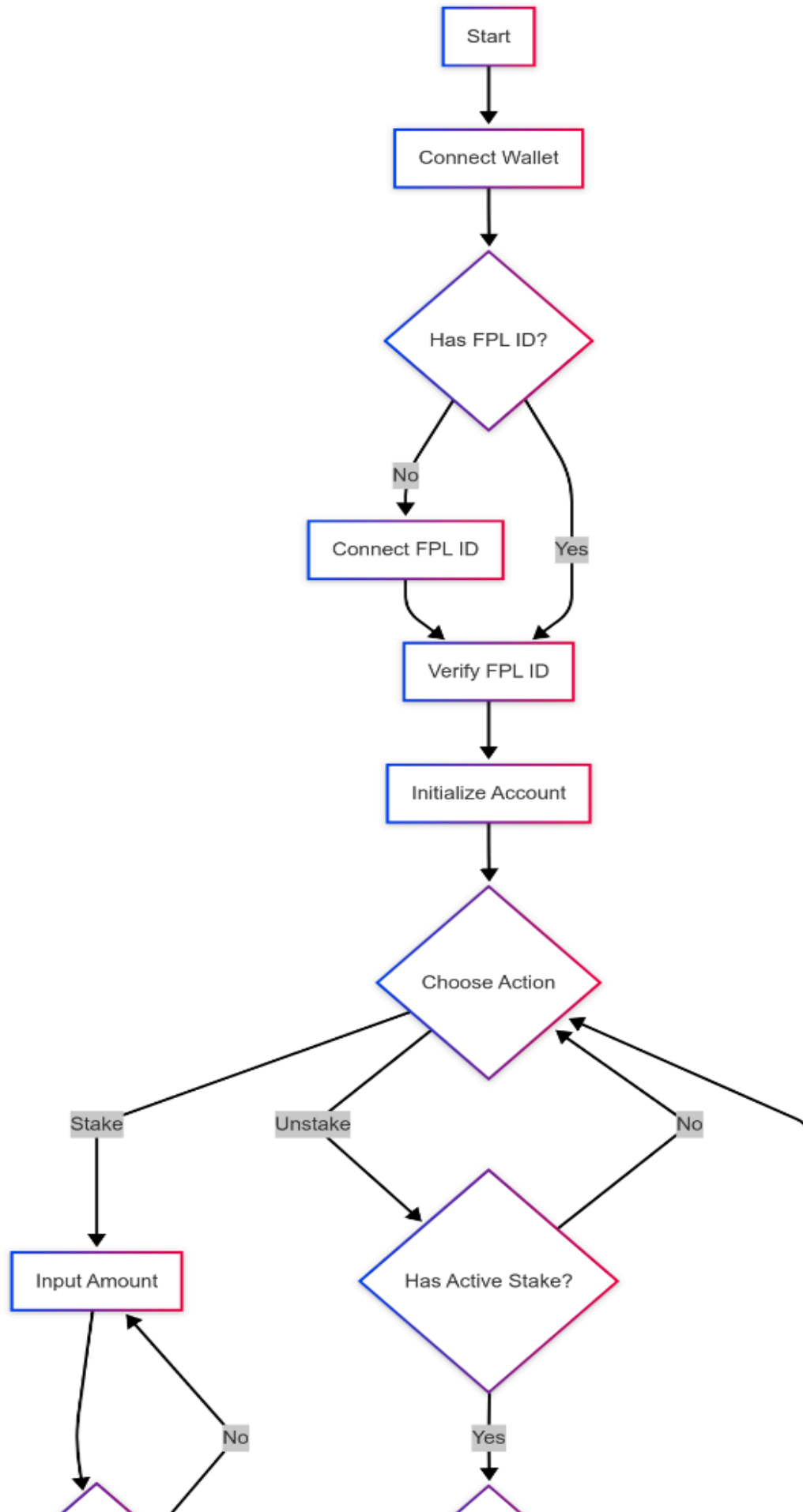Fig. 1.0 Flowchart of the software with the Unstake Feature.

```
                          ┌─────────┐
                          │  Start  │
                          └─────────┘
                               │
                               ▼
                       ┌───────────────┐
                       │ Connect Wallet │
                       └───────────────┘
                               │
                               ▼
                          ◇─────────◇
                         ╱           ╲
                        ╱  Has FPL ID? ╲
                        ╲             ╱
                         ╲           ╱
                          ◇─────────◇
                      No  │         │  Yes
                          ▼         │
                 ┌───────────────┐  │
                 │ Connect FPL ID │  │
                 └───────────────┘  │
                          │         │
                          ▼         ▼
                       ┌───────────────┐
                       │  Verify FPL ID │
                       └───────────────┘
                               │
                               ▼
                       ┌───────────────┐
                       │ Initialize Account │
                       └───────────────┘
                               │
                               ▼
                          ◇─────────◇
                         ╱           ╲
                        ╱ Choose Action ╲
                        ╲             ╱
                         ╲           ╱
                          ◇─────────◇
              Stake    ╱      │Unstake    ╲  No
                      ╱       ▼            ╲
           ┌───────────────┐  ◇─────────◇
           │ Input Amount  │ ╱           ╲
           └───────────────┘╱ Has Active  ╲
                  │   ▲     ╲   Stake?    ╱
                  │   │ No   ╲           ╱
                  ▼   │       ◇─────────◇
                             │ Yes
                             ▼
```
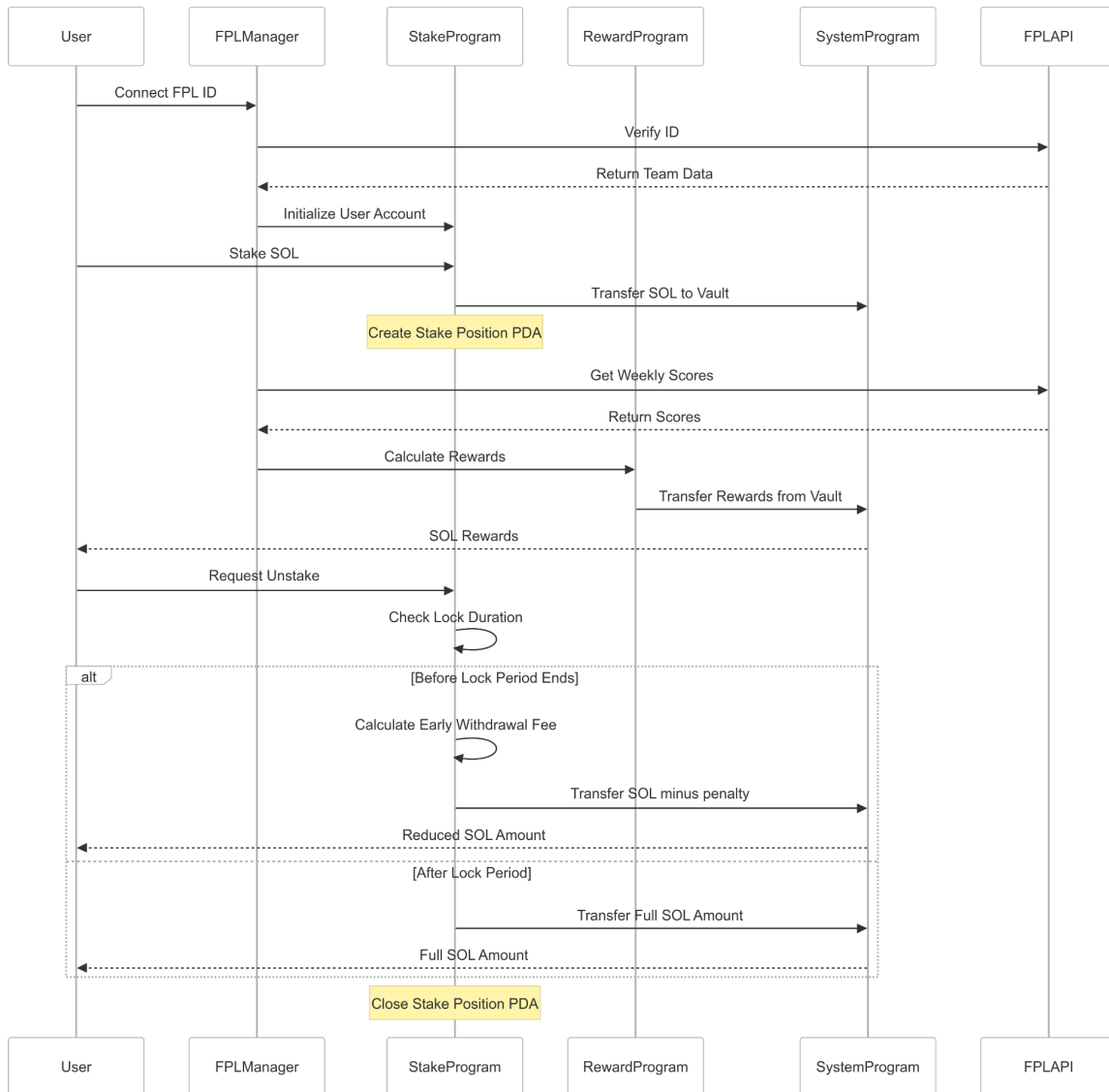
# Fig. 1.1 Sequence Diagram of the software with the stake Feature



**Account Structure.**

Fig 2.1 Class Diagram for the account structure, showing properties

## UserProfile

+Pubkey owner
+String fpl_id
+u64 active_stakes
+u64 total_rewards
+i64 last_sync
+[i32;38] performance_history
+u32 rank
+i64 join_date

+createProfile()
+updateFPLData()
+syncScores()

*reads config* → *creates* →

## GlobalState

+Pubkey authority
+Pubkey fee_account
+u64 total_stakes
+u64 total_rewards_distributed
+u8 penalty_rate
+u64 min_stake_duration
+bool is_paused
+i64 last_maintenance

+initialize()
+updateConfig()
+pauseProtocol()

## StakePosition

+Pubkey owner
+u64 amount
+i64 start_time
+u64 duration
+u64 accumulated_rewards
+i64 last_reward_calculation
+u8 performance_multiplier
+bool is_locked
+i64 withdrawal_window

+createStake()
+calculateRewards()
+processWithdrawal()

*manages* → *stakes in* → *configures* → *pays fees to* →

## TreasuryVault

+Pubkey authority
+u64 total_deposits
+u64 available_rewards
+u64 locked_stakes
+u8 reserve_ratio

## FeeVault

+Pubkey authority
+u64 total_fees_collected
+i64 last_withdrawal
+u8 fee_rate
+[Pubkey;3] fee_beneficiaries