# OasisBet Sport Betting Platform Functional Specification Document

## Project Objective

OasisBet aims to develop a secure and reliable online sports betting platform that provides users with an intuitive and seamless betting experience. The platform will offer a wide range of betting options for sports such as EPL, NBA, etc. The project objective is to create a user-friendly interface that allows users to easily deposit and withdraw funds, view available sports and odds, and place sport bets with confidence. The platform aims to provide users with other information such as valuable betting statistics, top picks for the week to enhance their betting experience. Ultimately, the goal of this project is to establish OasisBet as a trusted and established platform for online sports betting.

## Technical Objective

OasisBet aims to develop a high-performing, scalable, and secure microservice-based online sports betting platform using Angular, Java, and Spring Boot.

## Broad Business requirements

* Allow users to register and log in to the application.
* Allow users to view the list of available sports and odds.
* Allow users to place bets on various sports events.
* Allow users to update account details.
* Allow users to view account transactions.
* Allow users to deposit and withdraw money from their account.

## Microservices:

* Account Microservice: This microservice will handle user authentication, authorization, and registration. It will also handle user account management such as viewing transaction history and managing account settings.
* Betting Microservice: Responsible for managing and providing sports odds. This microservice would retrieve and store odds data from external sources, as well as provide this data to other microservices that require it.
* Result Microservice: Responsible for retrieving and storing sports results for completed events. This microservice would provide information on final scores, winning teams, and player stats, as well as schedules for upcoming events.

## APIs:

* Account Management API: This API will allow users to register, log in, and manage their account information.
* Betting API: This API will allow users to place bets on various sports events and view the odds and sports information.
  + <https://the-odds-api.com/>
  + <https://the-odds-api.com/liveapi/guides/v4/#overview>
  + <https://api.the-odds-api.com/v4/sports/?apiKey=4cb61e84b1c798ec69e03a836c068cea>

<https://api.the-odds-api.com/v4/sports/soccer_epl/odds?apiKey=4cb61e84b1c798ec69e03a836c068cea&regions=eu&markets=h2h&dateFormat=iso&oddsFormat=decimal&bookmakers=williamhill>

* Result API: This API will allow users to view sports events results and statistics.

## Architecture:

* The microservices will be deployed using a container orchestration system, such as Kubernetes.
* Each microservice will communicate with each other through REST APIs.
* The user interface will be built using Angular, which will consume the APIs provided by the microservices.

## Technology stack:

Front End - Angular will be used for building the user interface.

Back End - Java will be used for building the microservices. Spring Boot will be used as the framework for building the microservices.

Database - The databases used by the microservices will be relational databases, MySQL

## User Stories:

Account Management Microservice:

1. As a new user, I want to be able to create a new account on the betting platform with my personal information and login credentials.
2. As an existing user, I want to be able to login to my account using my username and password.
3. As a user, I want to be able to view my account details, including my current balance and transaction history.
4. As a user, I want to be able to change my password to ensure the security of my account.
5. As a user, I want to be able to change my contact information such as email and contact number.
6. As a user, I want to be able to view my current balance and betting limits.
7. As a user, I want to be able to withdraw any remaining balance in my account.
8. As a user, I want to be able to deposit money to my account.

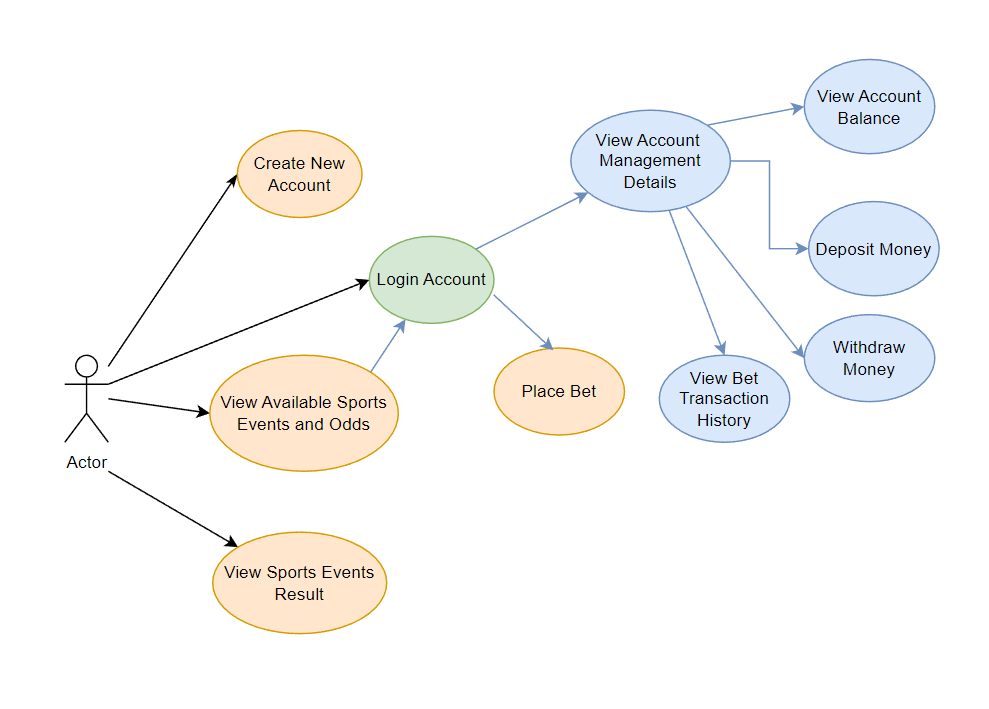
Betting Microservice:

1. As a user, I want to be able to view the available sports and events to place my bets on.
2. As a user, I want to be able to view the odds of each bet.
3. As a user, I want to be able to place a bet on a specific sport and event.
4. As a user, I want to be able to view my betting history, including all the bets I have placed, the amounts I have wagered, and the outcomes of those bets.
5. As a user, I want to receive notifications when the event that I placed a bet on has started and when the outcome is available.

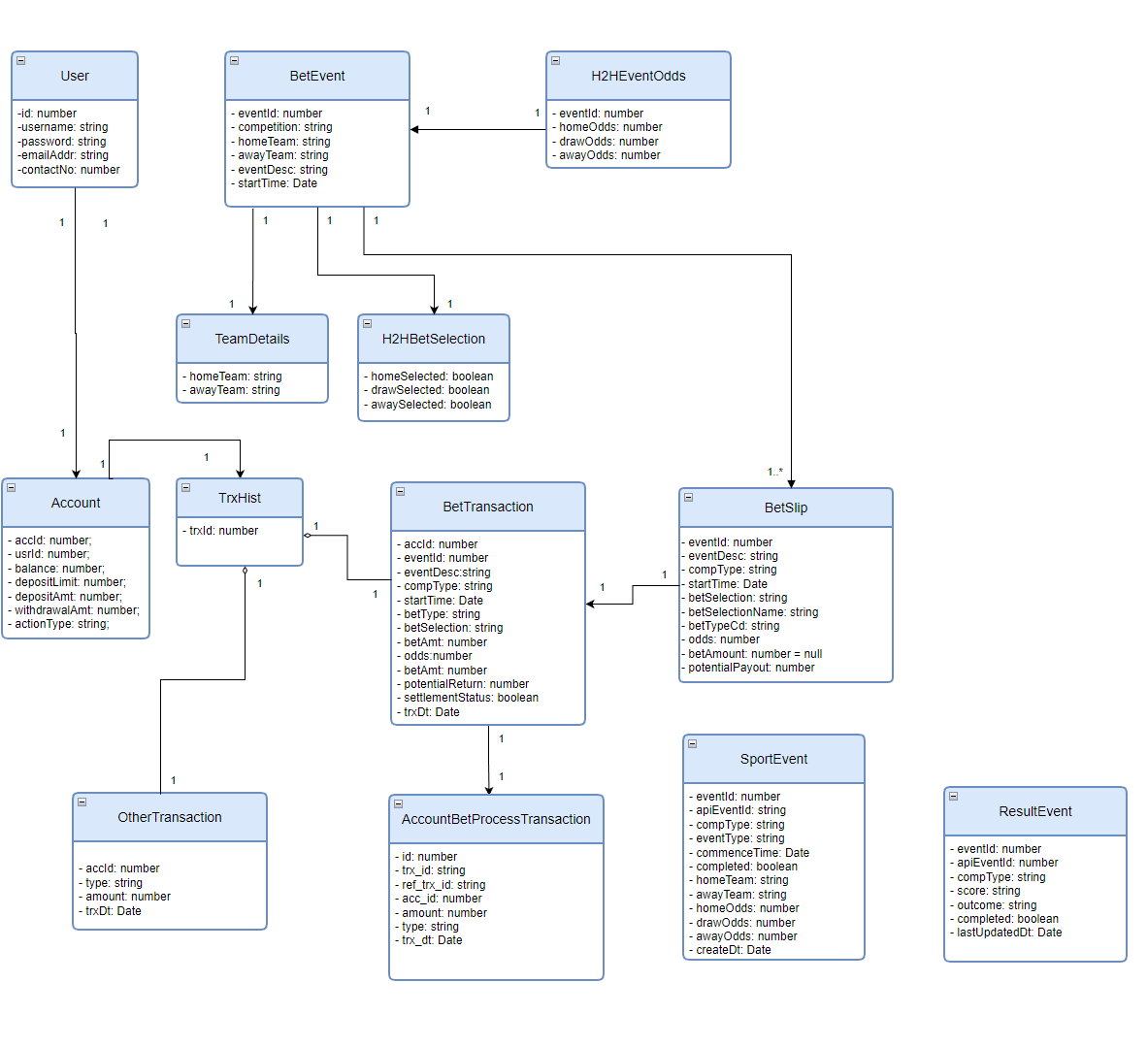
Result Microservice:

1. As a user, I want to be able to view the results of past matches.
2. As a user, I want to be able to search for match results based on team, date, or competition.
3. As a user, I want to be able to view the upcoming matches and their scheduled dates.
4. As a user, I want to be able to view the live scores and updates of ongoing matches.

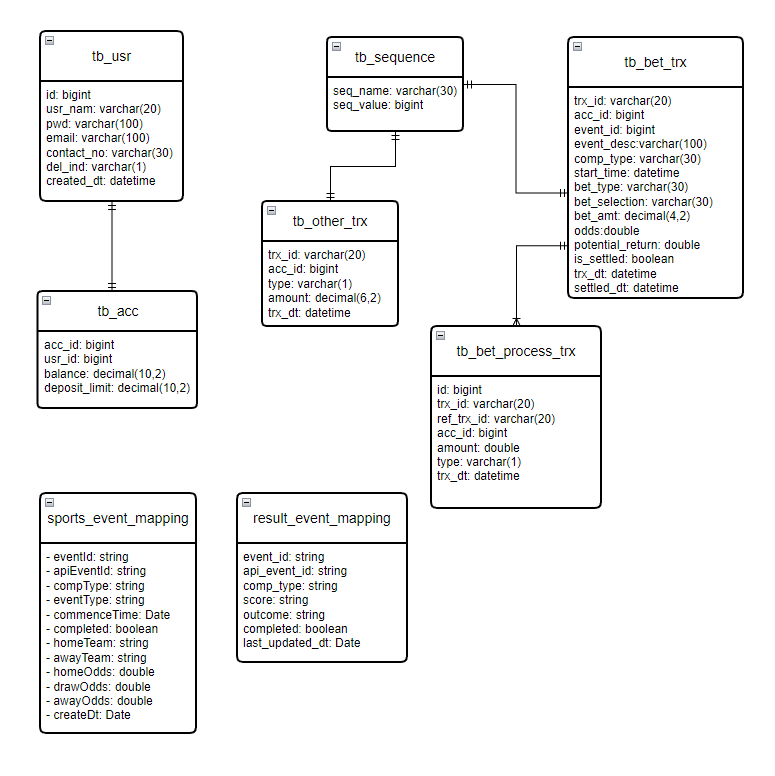
## Use Case Diagram:



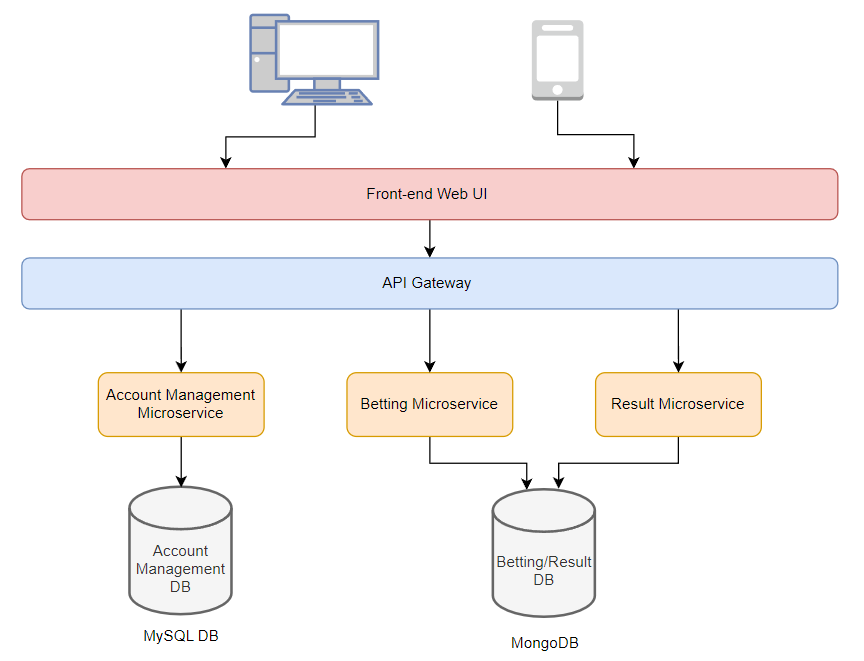
## Class Diagram:



## ER-Diagram:

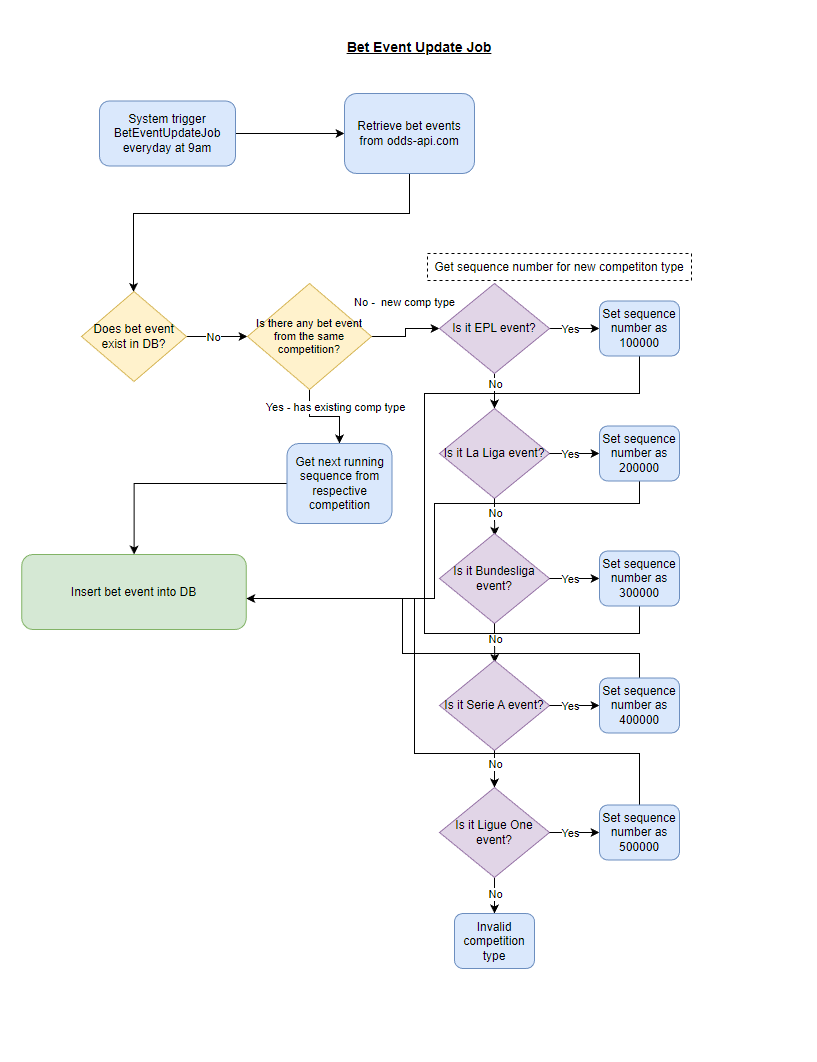


## Microservice Architecture Diagram:



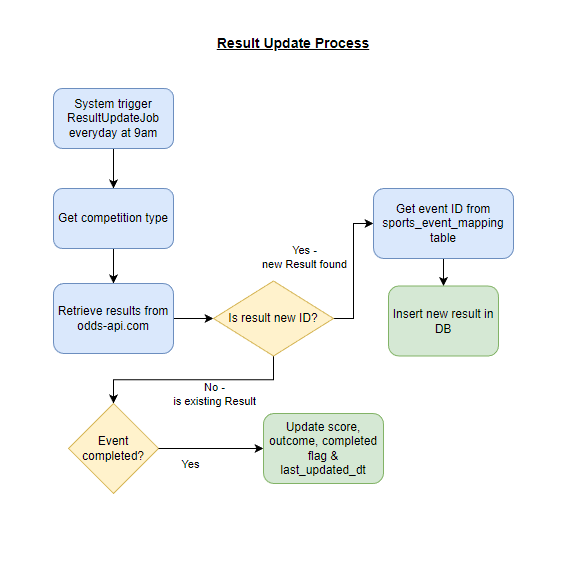
## Bet Event Update Job

Job will be triggered everyday at 9am to trigger the updating of the latest current bet events. New bet events will be inserted into the sports\_event\_mapping table in Mongo DB. A unique system generated Event ID will be assigned to every new bet event. The Event ID generated will be the running sequence from their respective competition.



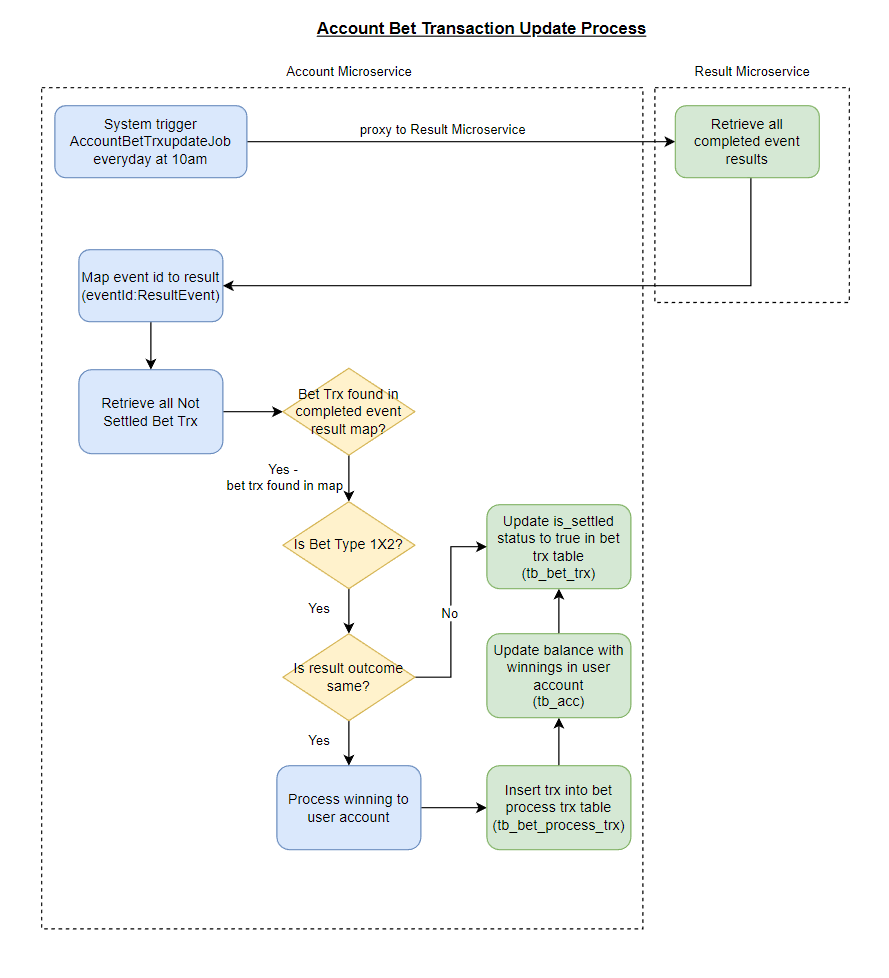
## Result Events Update

Job will be triggered everyday at 9am to trigger the result events syncing to update the available results. Results retrieved will be from 3 days ago to current date and time.



## Account Bet Transaction Update

Job will be triggered daily at 10am update bet transactions against the result of events. The event results will be retrieved from the Result Microservice via the Result Proxy in Account Microservice. All unsettled bets will be checked against the result events that were completed. Bet winnings will be credited and updated into user accounts.



## Login Authentication Sequence

A diagram of a software company

Description automatically generated

1. Client shall request server to generate JWT upon login, with client’s credential <username> and <password> in JWTTokenRequest object

HTTPS Endpoint to generate the JWT is https://[domain name]/user/authenticate.

1. Auth Server, which is in Account Microservice Server, authenticates the credentials against the Account Microservice Server DB, and if they are valid, a JWT, with an expiration time of 30mins is generated in the body of the response.
2. For subsequent requests, the generated token in the authorization header (type: Bearer token) must be sent.

Note: If a HTTP Status Code of 401 is obtained, this most probably means that

[1] The JWT token claims are invalid, and Step 1 needs to be repeated OR

[2] The credentials are not valid.

## JWT Refresh Token Sequence

A diagram of a project

Description automatically generated

1. When Client sends a request, Server shall validate JWT is expired. If JWT token is expired, a HTTP Status Code of 401 and error message of “Access Token Expired” is obtained.
2. Client shall request a for a JWT token refresh, with the existing expired JWT token in authorization header (type: Bearer).

HTTPS Endpoint to refresh the JWT is https://[domain name]/user/refreshToken.

1. Auth Server, which is in Account Microservice Server, authenticates the JWT token Claims, and if they are valid, another new JWT token, with an expiration time of 30mins is generated in the body of the response.
2. New JWT token will be set to replace the existing JWT token in the session storage of client side.
3. For subsequent requests, the new JWT token in the authorization header (type: Bearer token) will be sent.