Universidad de los Andes

Documentation & development report

ISIS 4226: Al for Software Engineering Project

Project - BetSmart

Members:

- Jose David Florez Ruiz
- Sebastian Gaona Castellanos
- Nicolás Ruiz Pérez
- German Andrés Sanchez Sarmiento

1. Description

BetSmart is an online sports betting platform designed to allow users to place bets on various sports events. The platform will include user registration, event listings, odds management, and bet placement. Users will be able to view live scores and track their betting history. The system aims to provide a seamless and engaging betting experience with secure transactions and user-friendly interfaces.

2. Objectives

- Develop a user-friendly sports betting platform
- Implement user authentication and account management
- Provide real-time event listings and betting odds
- Enable secure bet placement and transaction processing
- Provide a recommendation/insight to the user

3. Github Repository

- https://github.com/Project-AI-for-Software-Engineering/AI4SE-Project

4. Feature Distribution

Feature	Contributor(s)
Registration and Login	Sebastian Gaona
Profile Management	Sebastian Gaona
Password Recovery	Sebastian Gaona
Event Listings	Nicolás Ruiz
Odds Management	Nicolás Ruiz, Jose David
Event Filtering	Nicolás Ruiz
Bet Placement	German Sanchez
Transaction Management	German Sanchez
Bet Confirmation and History	German Sanchez
Live Scores	Jose David Florez
Result Notifications	Nicolas Ruiz, Jose David
Insight/Recommendation	Jose David Florez

5. Reflection on non-AI related challenges and limitations

During the development process, one significant challenge we encountered was the limitation imposed by sports API. The API restricts users to a maximum of 100 requests per day, which proved to be a considerable constraint for our testing phase. This limitation often left us unable to fully test our code, as we frequently reached the request cap before completing our intended tests.

As a result, we had to carefully prioritize our testing scenarios and frequently plan our requests to make the most of the limited quota. This restriction not only impacted our testing efficiency but also extended the development timeline, as we had to find alternative ways to simulate or work around the limited data access. In the future, exploring APIs with more

generous usage limits or implementing caching strategies to minimize API calls could help mitigate these problems.

Another challenge we faced was related to managing our code repository. We encountered difficulties with merging code changes, which often led to conflicts and delays. Additionally, we struggled with committing large files, such as the node_modules directory from React, which complicated version control.

Another significant challenge was related to our email notification system. We discovered that the system lacked the capability to schedule emails to be sent at a precise date and time. As a result, notifications were sent immediately upon the occurrence of certain events, such as viewing bets. While this approach provided some level of notification functionality, it did not meet our initial requirements for timed alerts, thereby limiting its effectiveness as a proof of concept. Moving forward, finding an email service with robust scheduling capabilities will be crucial to meet our notification needs.

Furthermore, we faced issues with our integration and unit testing. Due to prioritizing the development of main features, we couldn't allocate sufficient time to create and execute a comprehensive suite of tests. This oversight meant that some bugs and issues weren't identified until later stages of development, increasing the time required for debugging and fixes. In the future, ensuring that adequate resources and time are dedicated to testing from the outset will be essential to maintaining code quality and reducing development delays.

6. Reflection on how AI tools were used, and lessons learned:

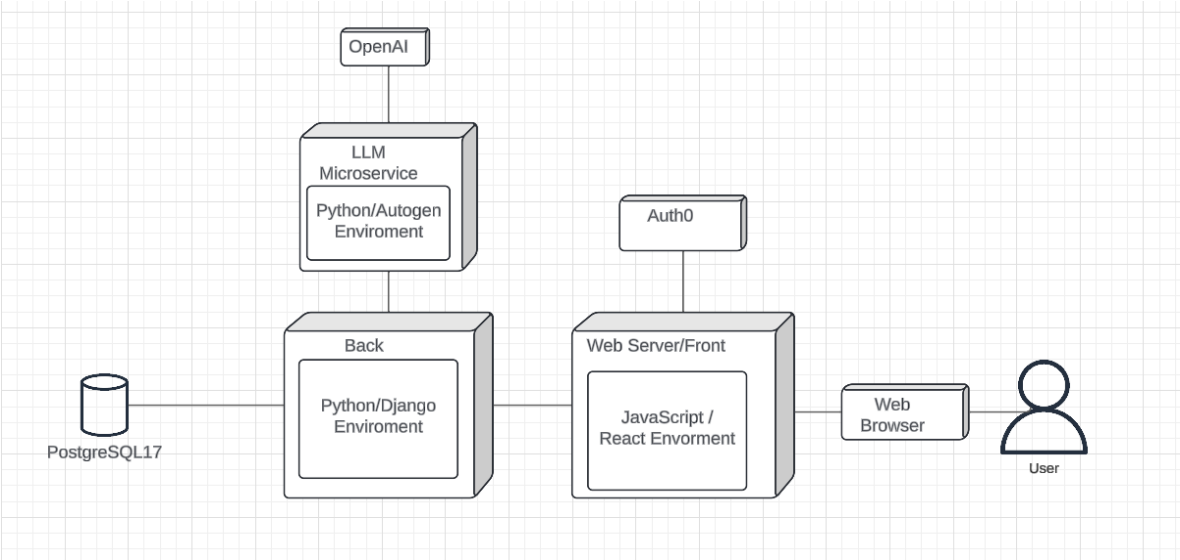
During the development process, one significant challenge we encountered was the limitation imposed by sports API. The API restricts users to a maximum of 100 requests per day, which proved to be a considerable constraint for our testing phase. This limitation often left us unable to fully test our code, as we frequently reached the request cap before completing our intended tests.

As a result, we had to carefully prioritize our testing scenarios and frequently plan our requests to make the most of the limited quota. This restriction not only impacted our testing efficiency but also extended the development timeline, as we had to find alternative ways to simulate or work around the limited data access. In the future, exploring APIs with more generous usage limits or implementing caching strategies to minimize API calls could help mitigate these problems.

Another challenge we faced was related to managing our code repository. We encountered difficulties with merging code changes, which often led to conflicts and delays. Additionally, we struggled with committing large files, such as the node_modules directory from React, which complicated version control.

7. System architecture and design

Deployment Diagram



8. User guides and API documentation

User guide:

1. Log In or Register

- Open the application.
- Click the "Login" button.
- Log in by entering your credentials (username and password) or, if you are a new user, register by filling out the registration form with your personal details.

2. View Available Matches

- After logging in, you have the option to view available matches.
- Click the "See Matches" button to view the list of all available matches.

3. Filter Matches

- To find specific matches, you can use the available filters.
- Select one of the filters
- A list of matches that meet the filter criteria will be displayed.

4. Place a Bet

- Click the "Bet" button next to the match you want to bet on.

- Select the match you want to place a bet on.
- Enter the winning team you believe will win.
- Specify the amount of money you want to bet.
- Once you place your bet, you will receive a confirmation email with the details of your bet.

5. View Your Bets

- To view your past bets, navigate to the "My Bets" section.
- Click the "My Bets" button to see a list of all your past bets.
- You will be notified when a match is finished, providing you with the result of your bet.

6. Update User Information

- If you need to change your personal information, you can easily do so.
- Click the "Update Profile" button.
- Fill in the fields with the new information.
- Save the changes to update your profile.

7. Log Out

- When you want to log out of your account, simply click the "Logout" button.
- This will securely log you out and return you to the login screen.

API documentation:

Sports-API used endpoints:
1. Endpoint for obtaining fixtures for one date:
Route: https://v3.football.api-sports.io/fixtures
Body Parameters:
{
'x-rapidapi-host' : 'v3.football.api-sports.io'
'x-rapidapi-key': The API key for Football API
}
Path Parameters:
Date: The desired date to know all matches.
2. Endpoint for obtaining a team´s statistics:
Body Parameters:
{
'x-rapidapi-host' : 'v3.football.api-sports.io'
'x-rapidapi-key': The API key for Football API

}
Path Parameters:
teamId: The id of the team
league: The id of the league
season: Always current season

Back API endpoints:

1. Endpoint for creating the prediction for a match:
Route: /api/v1/match-analysis/
Body Parameters:
{
"team1_id": the home team id,
"team2_id": the away team id,
"league": the match league id
}
2. Endpoint for creating a mail request in the database:
Route: /api/v1/create-mail/
Body Parameters:
{
“email”: It is the email address of the receiver
“time”: The time the mail is supposed to be sent
“msg”: The message for the email
}
3. Endpoint for sending all mails that the systems should send at that time:
Route: /api/v1/send-mails/
Body Parameters: Not required

4. Endpoint to get fund the wallet of a given user (1 in the example)
api2/wallets/1/recharge/
Body parameters:
{"amount": 1050}
5. Endpoint to make a transfer and a bet
{sender: sender,
receiver: receiver,
amount:amount,
eventid: match[3],
home: match[0],
away: match[1],
bet: selectedOption
}
6. Endpoint to get the historical transactions and bets for a given user

Api2/wallets/user_id:/history/

9. Test results and reports

Unit Tests for the Back endpoints:

```
PS C:\Users\ASUS\Desktop\AI4SE-Project\BackEnd>
----------------------------------------------------------------------
Ran 11 tests in 5.591s

OK
Destroying test database for alias 'default'...
PS C:\Users\ASUS\Desktop\AI4SE-Project\BackEnd>
```

```
PASS src/tests/Logout.test.js
  LogoutButton
    ✓ renders the logout button and calls logout on click (76 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.231 s
Ran all test suites matching /logout/i.
```

```
PASS src/tests/Login.test.js (5.575 s)
  LoginButton
    ✓ renders the login button and calls loginWithRedirect on click (136 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        14.734 s
Ran all test suites matching /login/i.
```

The Login test ensures the login button triggers the Auth0 login process correctly, while the Logout test confirms the logout button triggers the Auth0 logout process as expected. These tests check the functionality of individual components and their integration with Auth0.

The Update Profile test verifies API authentication and authorization by sending a request with a JWT token, ensuring the API processes the token, grants access to protected endpoints, and returns the correct data.

10. How to run the project

User Guide Backend | AI4SE-Project

How to run?

- 1. Open the folder 'Backend' as a project in vscode or other similar IDE.
- 2. Create a virtual enviroment with the command `python -m venv venv`.
- 3. Open the virtual enviroment you created `venv/Scripts/activate`.
- 4. Install all requirements with `pip install -r requirements.txt`.
- 5. Create a posgress data base called *_aise5_*, user *_postgres_*, password *_123_* .
- 6. Lunch the database.
- 7. Migrate database `python manage.py migrate`.
- 8. Migrate database `python manage.py makemigrations`.
- 9. Run the backend with the command `python manage.py runserver`.

- -

User Guide Frontend | AI4SE-Project

How to run?

- 1. Open the folder 'my-app' as a project in vscode or other similar IDE.
- 2. Make sure you have node.js installed.
- 3. Install dependencies with `npm install`.
- 4. Start the frontend with `npm start`.

Note: All dependencies are installed through the process.