Technical Specification

MMA Fantasy League

Members:

James Reilly - 19464192

Paul Bashford - 19709791

Supervisor: Mark Roantree

Date Completed: 24th February 2023

Table of contents

1. Introduction

1.1 Overview

Our MMA Fantasy league is a 24/7 cloud based web application. It is designed to give a platform to fans of the sport of Mixed Martial arts

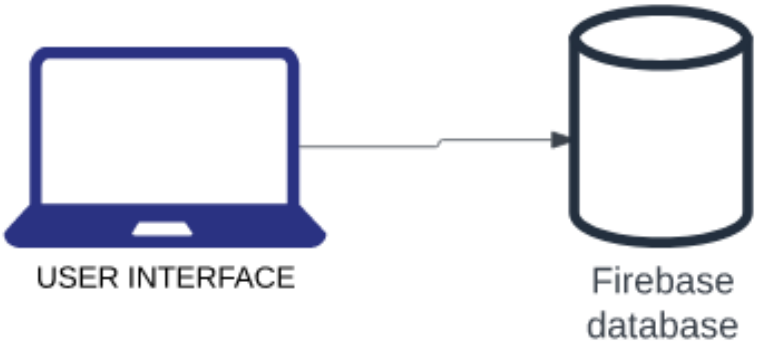
where they can compete against other users to try to predict the winners for each upcoming UFC event based on their knowledge of the sport. The app will take a number of different statistics into account that will be used to accumulate a point value for the UFC athletes as their likelihood of winning their upcoming fight. We're massive fans of multiple sports not exclusively including Mixed Martial Arts. The app is a full stack development and was made using django. The apps data is integrated with firebase, using its firebase database to hold authentication for users, the realtime database to hold information about user's league's and fighters and uses the firebase hosting to deploy our app

1.2 Glossary

Django: A high-level Python web framework for building web applications.
Firebase: A mobile and web application development platform developed by Google.
MMA: Mixed Martial Arts, a full-contact combat sport that combines techniques from various martial arts and combat sports.
Fantasy League: A type of online game where participants build a virtual team of real-life athletes and earn points based on the athletes' performance in actual matches.
Firebase Authentication: A Firebase service that provides backend services for user authentication and management.
Firebase Realtime Database: A cloud-hosted NoSQL database that allows users to store and sync data in real-time.

2. System Architecture

2.1. System Architecture Diagram



2.2 Overview

The system architecture for our Django project integrated with Firebase consists of a presentation layer, application layer, and data access layer. The presentation layer is responsible for rendering the views and templates in the front-end using our HTML , CSS, and JavaScript files. The application layer is the core of the Django project, handling user requests and responses through URL routing, views, and other python files used to send data to the database. The data access layer communicates with the real-time database of Firebase Firebase is used for authentication, real-time database, and cloud hosting.

2.3 Backend

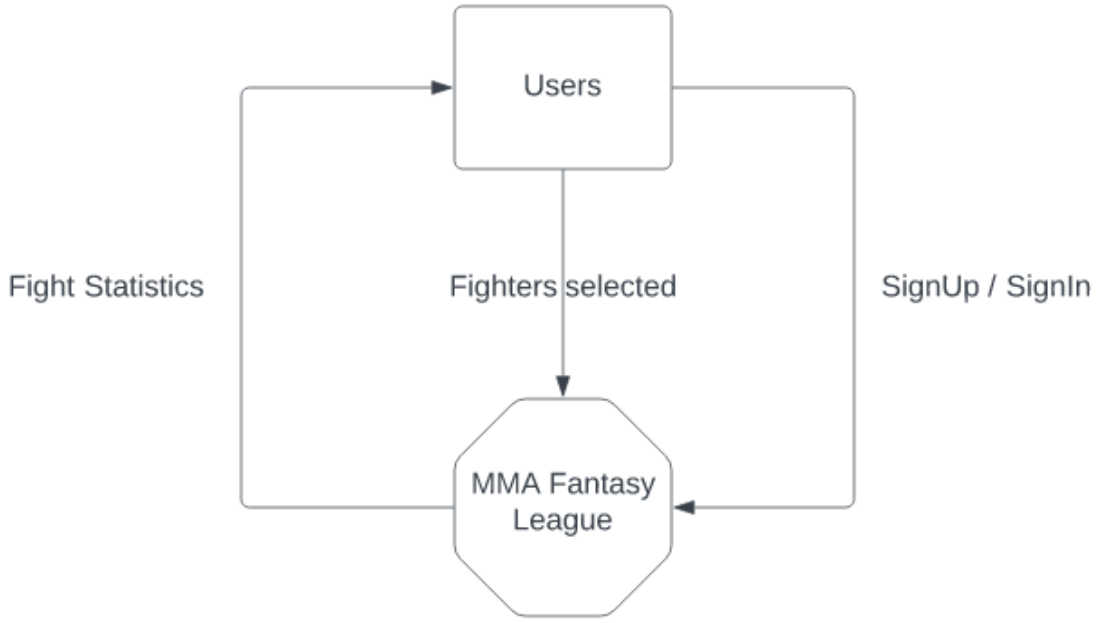
The backend of our database is deployed on firebase. We have integrated functions for our user sign up method that uses the firebase database authentication method to authenticate users. Creating an account a user is given a unique id (uid), based on a number made by their local machine ID. Using the user's email account they're saved in the database as a child called users, where users have children populated by each individual user of the app with the key being their email address. Firebase doesn't allow ley nodes to have '.' in them so we used the replace function .replace('.',',') to save the user email as the variable member_encoded_email.
The backend holds many functions like the fighter selection, create and join league, view league table and more. All these functions are hugely important to the functionality of the application.

2.4 Frontend

HTML is the basic structure of the web app, CSS styles it and controls its appearance, and JavaScript adds interactivity and dynamic behaviour to it.
To display the fantasy MMA site , we used HTML to structure the page and define content elements, CSS to style it and ensure it looks good across different devices, and JavaScript to add interactivity. These technologies work together to create an engaging and visually appealing website for fantasy MMA fans.

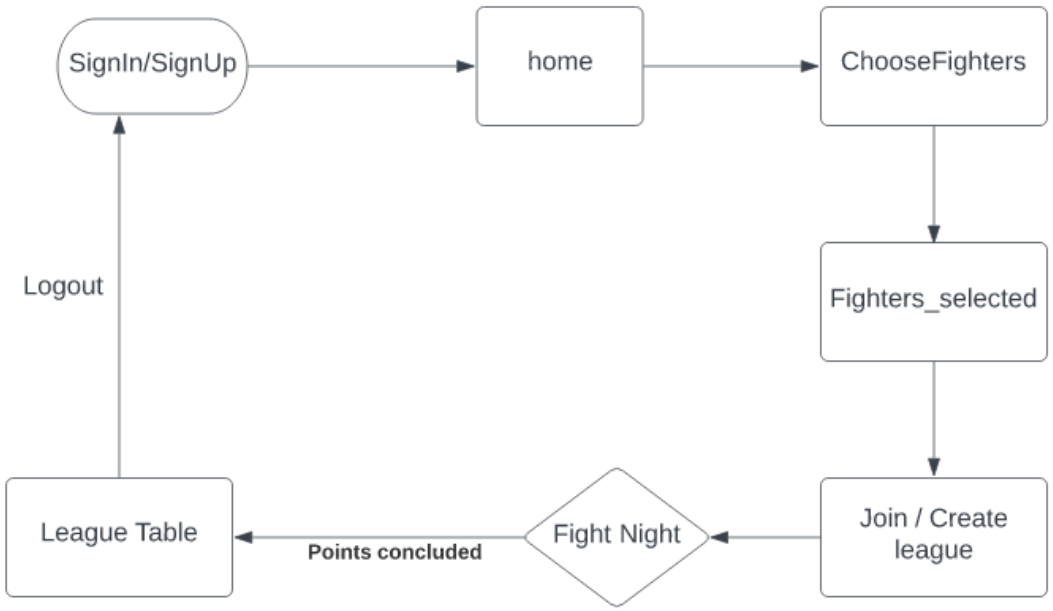
3. High-Level Design

3.1 Context Diagram



This context diagram for our MMAfantasy league shows the interactions between the web app and users with which the
system is designed to interface. It shows the information received by the web application and returned to the user.

3.2 Flowchart Diagram



This flowchart diagram displays a possible users path through the website where they sign in or sign up, go to the home screen, choose their fighters, view the fighters selected, join or create a league to compete against other users, then fight night occurs and the points are concluded and then shown on the league table in order highest points earned by each member .This flowchart can help to provide a clear understanding of the user's journey through the app, from the moment they log in until they receive their total points.

4. Problems and Resolution

4.1 Fantasy League Format

In the beginning we thought that the format for the fantasy league would be easy to develop a fantasy league logic system we had used many other fantasy leagues before but we quickly realised that it wouldn't be a simple as we had thought as the sport of MMA doesn't translate as easily to the Fantasy League format as some other sports. There are also no other MMA Fantasy leagues on the market so we had to invent the rules for it completely from scratch.

Solution: Essentially it was feedback from user testing. To start off we had intended to only allow users to select 5 fighters for their roster but after we got the feedback from our testing we had an overwhelming amount of testers suggest that we allow them to select a winner from each fight taking place on the night.

4.2 Data Integration

A large challenge in developing the fantasy league app was integrating real-time data with the app. It was both of the project members' first time dealing with a cloud database and we had particular difficulty getting the data to upload to the database and to pull relevant data from the database when it was required.

Solution: There was no quickfire solution to this problem. Mostly it involved extensive research and training on cloud databases and real-time data integration.

4.2 Testing

As this was our first database project of this scale we had to figure out how to design unit tests and how to get them to interact with the database. We found this particularly frustrating when the tests were not working properly but still uploading data to the cloud server which left the database quite unorganised

Solution: Once again it was an experience issue. We had to conduct extensive testing and research to understand how to properly get our tests to run. We found the Magicmock module particularly useful

as it allowed us to create mock data that wouldn't clog up our database while testing.

5. Installation Guide

To start off you'll need to ensure that you have Git and Python installed on your machine. Depending on your operating system you will need to open your command prompt or your terminal.
Windows: To open the command prompt, search for cmd in the search box. When you find cmd in the results, right click and select Run as administrator. This will open your command prompt.
Linux: To open the terminal, search for terminal in the search box. When the terminal appears in the results, click on it. This will open your terminal.
Download
Next you need to download the project files. The project can be downloaded at this link:
https://gitlab.computing.dcu.ie/reillj38/2023-ca326-bashfop2-reillj38
Alternatively you can download the repo from your terminal by inputting the following:
git clone https://gitlab.computing.dcu.ie/reillj38/2023-ca326-bashfop2-reillj38
Press enter then follow the instructions on screen to continue.

Setup

Once you have cloned the repo you have to navigate to where the files are stored on your
machine. This can be done by using the cd command **eg: cd 2023-ca326-bashfop2-reillj38**
Once you have successfully entered the directory you can move on to the next step.
Next you have to ensure that you have Django installed. You will also need to download the requirements for the app. To install Django or to check you have the most recent version
type into your terminal:
pip install django
and press enter.
The application should now be ready to run.
Starting the application.
When you have finished setting up the application, you need to start the server to get it up and running.
To do so, type in the command:
python manage.py runserver
If the server has successfully launched you can visit the webpage easily by opening your browser and typing in **http://127.0.0.1:8000/.**