Leaguelizer

Documentation

by Sali Arnold

Contents

1.Introduction

1.1.Overview

Leaguelizer is a web application for managing fantasy football leagues. It incorporates 4 objects: stadiums, competitions, clubs and matches, statistical reports about league budgets, club stadium capacities. For handling the users, every user gets an account, with different privileges to manage the whole fantasy league or just specific elements.

1.2.Goal

The goal of the application is to provide an easier platform where users can manage every aspect of a fantasy football league. For the sake of better management users are also able to manage user specific items.

1.3.Starting the application

1.3.1.Development Environment

To start the development environment the following steps need to be followed:

1. Install Docker on the machine
2. Install Node.js on the machine
3. Create a .env file in the backend folder of the application with the following fields:

```
DB_ENGINE=django.db.backends.sqlite3
DB_NAME=db.sqlite3
DB_USER=none
DB_PASS=none
DB_HOST=none
DB_PORT=none
```

4. Start the backend with the following command:

```
docker-compose-development up -d --build
```

5. In the frontend/leaguelizer/src/Context/Context.js file change the URL_BASE and URL_SOCKET variables to localhost:8000

```
const URL_BASE = "https://localhost:8000";
const URL_SOCKET = "wss://localhost:8000/ws/room/";
```

6. Run the frontend with the following command:

```
npm start
```

The application is now running on localhost:3000

1.3.2.Production Environment

To start the production environment the following steps need to be followed:

1. Install Docker on the machine
2. Create a postgres database in another machine(ex: google cloud databases)
3. Create a .env file in the backend folder of the application with the following fields:

```
DB_ENGINE=django.db.backends.postgresql
DB_NAME=leaguelizer
DB_USER=db_user
DB_PASS=db_user_123
DB_HOST=34.116.177.169
DB_PORT=5432
DEBUG=0
VIRTUAL_HOST=SArnold-sdi-22-23.crabdance.com
VIRTUAL_PORT=8000
LETSENCRYPT_HOST=SArnold-sdi-22-23.crabdance.com
DEFAULT_EMAIL=saliarnold2000@gmail.com
NGINX_PROXY_CONTAINER=nginx-proxy
DJANGO_ALLOWED_HOSTS=SArnold-sdi-22-23.crabdance.com
```

- The host variables are the url for the backend
- The email address is for the ssl certificate(it sends an affirmation to the email)
- The db setting:
  - DB_NAME: the name of the database
  - DB_USER: username for accessing the postgres database
  - DB_PASS: password for accessing the postgres database
  - DB_HOST: ip address where the postgres database is located(ex: google cloud)
  - DB_PORT: port in which the postgres database can be accessed

4. Run the backend with the following command:

```
docker-compose up -d --build
```

5. In the frontend/leaguelizer/src/Context/Context.js file change the URL_BASE and URL_SOCKET variables to the url of the backend (the one you used at the host variables above)

```
const URL_BASE = "https://<URL>";
const URL_SOCKET = "wss://<URL>/ws/room/";
```

6. Deploy the frontend on Netlify directly from the github repository, with the following build command: npm run build

2. About the application

2.1.Key Features

2.1.1.Managing the league

Every user can manage the clubs, stadiums, competitions and matches in the league depending on their user privileges. A standard user can manage their own instances, meaning they can create new ones, change and delete their own ones. A moderator can manage all the instances in the league, just like an admin.Having a centralized application to handle all of these actions, make it easier to handle the whole league, and keep track of everything.

2.1.2.Account Management

The registering allows a user to engage with the league, since creating anything in the league requires an account. The login is based on a secure email and password authentication, which allows a user to manage his own objects in the league, and see his own account. The normal user has a standard account, getting moderator or admin privileges is up to the admin user.

2.1.3.Statistical Reports

The application generates statistical reports about the budget of different leagues, and the stadium capacity of the different clubs. These reports are publicly available and help the users to understand the details of the league much better.

2.1.4. Profile View

Users have access to their personal profiles, where they can view and update their information. This includes details such as a bio, location, birthday, gender, marital status and number of instances they own from clubs, stadiums, competitions and matches. Profiles enhance the social aspect of the app, allowing users to connect and share their travel experiences.

2.1.5. Admin View

Administrators have a dedicated view in the application, in which they delete elements in bulk. This feature provides a more easier way to manage the application and keep everything in track. In this view they can also manage the different users, changing their role and the amount of elements the user can see in a page. The admin view provides a way to manage both the league and the users of the application in a more convenient fashion.

2.1.6. Chat

The application provides a global anonymous chat for all users. The users can freely choose their nickname to talk with other people in the application.

3.Backend Technologies

3.1. Django

Django is a Python-based web framework that provides developers with a structured and efficient approach to building web applications. It follows the model-view-controller (MVC) architectural pattern, allowing for the separation of concerns and promoting code reusability. With its built-in features for database management, URL routing, and user authentication, Django enables the development of secure and scalable web applications in a shorter timeframe. The downside of choosing this framework is the fact that python is a dynamic language, which means it is slower compared to something like C++.

3.2.DRF

Django REST Framework (DRF) is a powerful toolkit that extends the capabilities of Django for building robust and scalable Web APIs. It provides a comprehensive set of tools and libraries for quickly developing RESTful endpoints, handling authentication and permissions, and serializing data in various formats like JSON and XML. With its built-in support for authentication schemes, throttling, and serialization, DRF simplifies the process of building API-driven applications with Django, making it a popular choice for developers.

3.3. Neural Networks

Neural Networks are a class of machine learning models particularly suited for predicting scores in multiclass problems. These networks consist of interconnected nodes, called neurons, organized in layers, and they excel at capturing complex patterns and relationships within the data. By training on known input-output pairs, neural networks can accurately predict scores for various classes, making them a powerful tool for applications like sentiment analysis, image classification, and recommendation systems in which multiple classes or categories need to be assigned scores. For the sake of simplicity the model is considered done, meaning it doesn’t learn on the fly.

3.4. Tensorflow

TensorFlow is a powerful open-source library for machine learning and deep learning tasks. It provides a comprehensive set of tools and functionalities for building and deploying various types of neural networks, enabling developers to create complex models for tasks like image recognition and natural language processing. TensorFlow's strengths lie in its flexibility, scalability, and extensive community support, allowing for efficient development and deployment of machine learning models across different platforms. However, its learning curve can be steep for beginners, and TensorFlow's complex architecture and setup requirements may pose challenges for those new to machine learning.

3.5. Swagger

Swagger is a powerful tool for documenting and testing RESTful APIs, providing a standardized format for describing endpoints, request/response formats, and authentication mechanisms using the OpenAPI Specification (OAS). Its main advantage is that it allows for the generation of interactive API documentation, enabling developers to easily explore and understand the API's capabilities. However, maintaining Swagger documentation can become burdensome if not kept up to date with the actual API implementation, and it may require additional effort to ensure the synchronization of changes between code and documentation.

3.6. Environment Variables

Environment variables are stored in every machine that runs the backend application. Its purpose is to store configuration-specific and sensitive information specifically. They also ensure flexibility by not having to re-write code for a change to occur.

3.7. Postgres

PostgreSQL, often referred to as Postgres, is a powerful and feature-rich open-source relational database management system. It offers robust data integrity, transactional support, and advanced features such as full-text search, geospatial support, and JSON document storage. PostgreSQL's extensibility and support for complex queries make it well-suited for handling large datasets and providing scalability. However, it may require more configuration and administration compared to simpler databases, and its performance can be impacted under heavy write workloads. Additionally, PostgreSQL's learning curve can be steep for beginners due to its extensive feature set and advanced functionality.

3.8. SQL Scripts

SQL Scripts generation with Faker and a SQL scripting language like Python offers a convenient way to populate databases with realistic and synthetic data. By utilizing the Faker library, developers can generate custom datasets for testing, development, or demonstration purposes, saving time and effort in manually creating data. Since the data is randomized, the use case of these data is the testing of the structural integrity of the system, and the testing of the optimization of the system

3.9. Faker

Faker is a versatile Python library that generates realistic and random data, often used for SQL script generation. It provides a wide range of data types and functions to create dummy data, such as names, addresses, dates, and numbers, making it useful for populating databases for testing or development purposes. One of the main advantages of Faker is its ease of use and flexibility, allowing developers to quickly generate custom data sets. However, it is important to note that Faker is not suitable for generating production data and should be used solely for non-sensitive or fictional data generation, as done in this case to test the durability and speed of the application.

3.10. Pagination

Pagination is a technique used in software development to divide large sets of data into smaller, more manageable chunks called pages. It enables efficient navigation and presentation of data, reducing loading times and improving user experience. However, pagination can introduce complexities when dealing with dynamic or filtered data, as it may require additional logic to handle sorting, filtering, and maintaining consistent pagination across various pages.

3.11. Autocomplete

Autocomplete is a functionality in software applications that provides suggestions or completions as a user types input into a search or input field. It improves user experience by saving time and reducing errors in data entry. However, implementing autocomplete may require additional development effort to integrate with data sources, and the accuracy of suggestions depends on the quality and relevance of the underlying data.

3.12. Consumers

Consumers in Django refer to the components that handle WebSocket connections and manage real-time communication between the server and the client. They enable the development of interactive and dynamic web applications by facilitating bidirectional communication. However, implementing consumers can be more complex compared to traditional HTTP views, and it requires support from a compatible WebSocket server. Additionally, scaling WebSocket connections and handling concurrency can pose challenges in high-traffic scenarios. Also changing the whole application from wsgi to asgi, having the need for an external server to run the application, namely Daphne.

3.13. Daphne

Daphne is a lightweight and high-performance HTTP and WebSocket server that is commonly used with Django applications. It provides an efficient and scalable solution for handling WebSocket connections and serving HTTP requests. However, Daphne may require additional configuration and setup compared to more traditional WSGI servers, and it may not be the best choice for applications with heavy traffic or complex deployment scenarios that require advanced load balancing or proxying capabilities.

3.14. JWT Tokens

JWT (JSON Web Tokens) is a widely used authentication mechanism in Django for securing API endpoints. It allows for stateless authentication by encoding user information into a compact and digitally signed token, eliminating the need to store session data on the server. However, JWTs have the drawback of being non-revocable, meaning once issued, they remain valid until they expire, requiring careful management of token expiration and potential token revocation scenarios. Additionally, JWTs may increase the size of requests as the token itself needs to be included with each authenticated request.

4. Frontend Technologies

4.1. React

React is a popular JavaScript library used for building user interfaces in web applications. It utilizes a component-based architecture that enables the creation of reusable and modular UI components, making development more efficient and maintainable. React's virtual DOM and efficient rendering process optimize performance by minimizing unnecessary updates. However, the learning curve for React can be steep for beginners, and integrating React into existing projects may require additional effort. Additionally, React focuses primarily on the view layer, so developers need to rely on external libraries or frameworks for additional functionalities like routing and state management.

4.2. Fetch

Fetch is a web API in JavaScript that provides a simple and flexible way to make HTTP requests to fetch resources from a server. Compared to the axios counterpart, it misses a lot of useful functionalities, but for the sake of not adding more complexity to the application I choose Fetch.

4.3. Responsive Design

Responsive design is an approach to web development that aims to create websites and applications that adapt and display optimally across various devices and screen sizes. It provides a seamless and consistent user experience by automatically adjusting layouts, images, and content based on the device's screen dimensions. The benefits of responsive design include improved accessibility, increased user engagement, and reduced development and maintenance efforts. However, implementing responsive design may require additional development time and complexity, especially when dealing with complex layouts or legacy systems. Balancing the design across different screen sizes can also pose challenges in preserving the desired user experience

4.4. Material UI

Material UI is a popular React component library that implements the Material Design guidelines, providing a set of pre-designed and customizable UI components for building modern and visually appealing web applications. It offers a wide range of components, themes, and styling options, saving development time and ensuring consistent and responsive designs. However, the extensive set of components and customization options may lead to increased bundle size and potential performance implications, and the learning curve for newcomers to Material UI can be steep due to the need to understand its specific API and styling conventions.

4.4.1. Flexible Table

I made a flexible Table module out of the Material UI table, which gets a specific list of dictionary elements, gets the headers and the content, and dynamically displays them.

Although It may seem that the preparation functions are not called in the given page, it actually reduces the multiplicity of the code across the application.

4.5. Toast

Toast is a user interface component commonly used in web applications to display temporary messages or notifications to the user. It provides a non-intrusive and visually appealing way to convey information or alerts without interrupting the user flow. The benefits of using toast include enhanced user experience, easy implementation, and the ability to provide timely feedback. However, excessive or poorly designed use of toast notifications can be disruptive and overwhelm users, potentially leading to information overload and distraction. Careful consideration should be given to the frequency, duration, and relevance of toast messages to strike the right balance.

4.6. JWT

JWT (JSON Web Tokens) is a widely adopted authentication mechanism that securely transmits user information between a client and a server as a digitally signed token. Its advantages include statelessness, allowing for scalable and distributed authentication, and enabling single sign-on across multiple systems. However, JWTs have a fixed expiration time and cannot be revoked before expiration, posing a challenge in scenarios where immediate revocation is necessary. Additionally, the token size can grow significantly when storing additional user data, potentially impacting network performance. Care should be taken to properly secure and handle JWTs to prevent token tampering or misuse.

4.7. Websockets

WebSockets is a communication protocol that enables real-time, bidirectional communication between a client and a server over a single, long-lived connection. It allows for instant data transmission and eliminates the need for frequent requests and page reloads, leading to faster and more interactive web applications. The advantages of using WebSockets include real-time updates, efficient data transfer, and enhanced user experience. However, implementing WebSockets can introduce complexity, especially in managing connection state and handling concurrency. Additionally, it requires support from both the server and client, limiting compatibility with older browsers and some server configurations. In the frontend this technology is used for the sake of a chat page, where users can communicate to each other. This page is limited to the last 10 messages for the sake of keeping the optimization in check.

5.Deployment technologies

5.1. Google Cloud

Google Cloud is a comprehensive cloud computing platform that offers a wide range of services and tools for building, deploying, and managing applications and infrastructure in the cloud. It provides scalable and reliable infrastructure, data storage, machine learning capabilities, and other services that enable developers to create robust and innovative solutions. The advantages of using Google Cloud include its global infrastructure, high availability, automatic scaling, and integration with other Google services. However, depending on the specific requirements, the cost of using Google Cloud services can vary and may be a consideration. Additionally, configuring and managing the various services within the platform may require familiarity with Google Cloud's documentation and learning curve.

5.2. VM

The app is deployed using Google Cloud Compute Engine, which provides virtual machines (VMs) for running the app's backend and other services. Firewall rules are implemented to control inbound and outbound network traffic to the VMs, ensuring secure access and protecting against unauthorized access or malicious activity.

5.3. Google Cloud Database Engine

Google Cloud Database Engine is a fully managed relational database service provided by Google Cloud. It offers scalable and highly available databases, including MySQL, PostgreSQL, and SQL Server. The advantages of using Google Cloud Database Engine include automatic backups, automated patch management, and seamless integration with other Google Cloud services. However, the pricing structure can be complex, and costs can increase with additional storage or high I/O usage. Additionally, some advanced database features may be limited compared to self-managed database solutions, and migration from on-premises databases or other cloud providers may require careful planning and consideration.

5.4. Docker

Docker is an open-source platform that simplifies the process of building, packaging, and deploying applications using containerization. It provides an isolated and lightweight environment for applications to run consistently across different platforms and environments. The advantages of using Docker include improved scalability, easier application deployment, and efficient utilization of resources. However, managing multiple containers and orchestrating them in complex environments can be challenging and may require additional tooling and expertise. Additionally, the overhead of containerization can introduce some performance overhead compared to running applications directly on the host system.

5.5. Gunicorn

Gunicorn with uvicorn is a popular combination for serving Python web applications. Gunicorn acts as a web server, managing worker processes to handle incoming requests efficiently, while uvicorn serves as the application server, providing an ASGI (Asynchronous Server Gateway Interface) implementation for handling asynchronous Python frameworks like FastAPI and Starlette. The advantages of using Gunicorn with uvicorn include their compatibility with a wide range of Python frameworks, support for handling high loads, and the ability to leverage the performance benefits of asynchronous programming. However, configuring and fine-tuning Gunicorn and uvicorn can be complex, and handling long-running requests or heavy I/O operations may require careful consideration to avoid blocking worker processes and impacting overall performance.

5.6. Nginx

Nginx is a powerful web server and reverse proxy server that excels at handling high traffic and concurrent connections. It is known for its efficient and event-driven architecture, which enables it to handle a large number of requests with low resource consumption. The advantages of using Nginx include its excellent performance, ability to handle simultaneous connections, and robust load balancing and caching capabilities. However, configuring and fine-tuning Nginx can be complex, especially for users unfamiliar with its configuration syntax, and advanced features such as HTTPS and SSL termination may require additional setup and certificate management.

5.7. Let’s Encrypt

Let's Encrypt is a free and automated certificate authority that provides SSL/TLS certificates for securing websites. It simplifies the process of obtaining and renewing certificates, allowing website owners to enable HTTPS encryption easily. The advantages of Let's Encrypt include its cost-effectiveness, automated certificate management, and compatibility with a wide range of web servers and platforms. However, Let's Encrypt certificates have a shorter validity period compared to some paid options, and wildcard certificates may require additional steps for verification. Additionally, Let's Encrypt may not be suitable for organizations with specific compliance or audit requirements that necessitate the use of specific certificate authorities.

5.8. FreeDNS

FreeDNS is a DNS hosting service that allows users to manage and map their domain names to IP addresses. It provides a free and convenient solution for individuals or small organizations to establish a web presence without the need for purchasing a dedicated domain name. The advantages of FreeDNS include its cost-effectiveness, easy setup process, and flexibility in managing DNS records. However, relying on a free DNS service may come with limitations such as fewer advanced features, potential limitations on DNS record types or update frequency, and possible advertisements. It is essential to assess the specific needs and requirements of a project before deciding to use FreeDNS.

5.9. Netlify

Netlify is a powerful web development platform that simplifies the process of deploying, hosting, and managing websites and web applications. It provides features such as continuous deployment, automatic SSL certificates, and CDN (Content Delivery Network) integration, making it easy to deploy and scale projects. The advantages of using Netlify include its simplicity, seamless integration with Git workflows, and the ability to deploy static websites quickly. However, Netlify's focus on static websites may limit the capabilities for dynamic server-side functionalities, and certain advanced features may require additional configurations or a higher-priced plan.

6.Testing and Performance Technologies

6.1. JMeter

JMeter is a popular open-source tool designed for load testing and performance measurement of web applications. It provides a user-friendly interface to create and execute test plans, simulate various types of requests, and analyze application performance under different loads. The advantages of using JMeter include its flexibility, extensibility through plugins, and ability to simulate realistic scenarios. However, configuring complex test plans and interpreting test results may require technical expertise. JMeter's resource consumption can also be significant, and generating high loads may require distributed test setups.

6.2. Cypress

Cypress is a JavaScript-based end-to-end testing framework designed to facilitate the development of reliable and robust web applications. Its key advantage lies in its ability to run directly in the browser, enabling real-time debugging and instant feedback during test execution. However, the framework's reliance on a single browser limits cross-browser compatibility and may lead to inconsistencies in test results, making it less suitable for projects that require comprehensive multi-browser testing.

6.3. Unit Test

Unit Testing is a software testing method where individual components or units of code are tested in isolation to ensure they function correctly. It focuses on verifying the functionality of specific units of code, such as functions or methods, by providing known inputs and comparing the actual output with the expected output. By automating the testing process and catching errors early on, unit tests help improve code quality, enhance maintainability, and provide confidence in the stability of software applications.

6.4. Google Cloud Statistics

Google Cloud Statistics is a powerful statistical analysis tool provided by Google Cloud Platform. It offers a wide range of statistical functions, algorithms, and visualization capabilities, enabling users to explore and derive insights from large datasets. The key advantages of Google Cloud Statistics include its scalability, seamless integration with other Google Cloud services, and the ability to leverage Google's infrastructure for high-performance computing. However, its complexity and learning curve might pose challenges for beginners, and it may have certain limitations compared to specialized statistical software in terms of advanced statistical techniques and customization options.

6.5. Matplotlib

Matplotlib is a versatile data visualization library for Python that provides a wide range of plotting functionalities. It offers a straightforward syntax and extensive customization options, allowing users to create visually appealing and publication-quality plots. Its pros include a vast array of plot types, compatibility with various data formats, and a large community contributing to its development. However, its learning curve can be steep for beginners, and creating complex visualizations may require additional effort compared to specialized visualization libraries.

7.Management Technologies

7.1. Clickup

ClickUp is a comprehensive project management and collaboration tool that helps teams streamline their workflows and stay organized. Its pros include a wide range of features such as task management, time tracking, document sharing, and integrations with other popular tools. With a user-friendly interface and customizable options, ClickUp promotes team collaboration and efficiency. However, some users may find the extensive feature set overwhelming, and certain advanced functionalities may require a learning curve for users new to project management software.

7.2. Git

Git is a distributed version control system widely used in software development to track changes in source code. Its key advantages include efficient collaboration, as multiple developers can work on the same project simultaneously and merge their changes seamlessly. Git also provides the ability to revert to previous versions and branches, making it easy to experiment and manage different versions of the codebase. However, its learning curve can be steep for beginners, and resolving merge conflicts can sometimes be complex and time-consuming, requiring knowledge of Git's command-line interface.

7.2.1. Feature Branching

Git Feature Branching is a software development workflow that involves creating separate branches for each new feature or enhancement. It allows developers to work on features independently without affecting the main codebase, providing isolation and reducing the risk of conflicts. The advantages of Git Feature Branching include improved collaboration, easier code review, and the ability to iterate on features without disrupting the stability of the main branch. However, managing a large number of branches can become cumbersome, and merging features back into the main branch may occasionally introduce conflicts that require careful resolution.

7.3. Github

GitHub is a web-based platform for version control and collaborative software development. It provides a centralized repository for storing and managing code, making it easy for developers to collaborate on projects and track changes over time. The key advantages of GitHub include its user-friendly interface, extensive community support, and integrations with popular development tools. However, the free tier of GitHub has certain limitations, such as restrictions on private repositories, and there is a learning curve associated with mastering its features and workflows.

7.4. UML Diagrams

UML (Unified Modeling Language) diagrams are a standardized visual representation used in software engineering to depict software systems and their components. They provide a common language for communication between developers, stakeholders, and designers, enabling better understanding and collaboration. The advantages of UML diagrams include improved clarity, documentation, and analysis of software systems, facilitating better design decisions and identifying potential issues. However, UML diagrams can sometimes become complex and time-consuming to create and maintain, and excessive reliance on diagrams can lead to an overemphasis on documentation rather than actual code implementation.