UAZ Rockety Design Doc

By: Kade Dean

2/20/2024

Table Of Contents

Purpose

The purpose of this website, a project sponsored by the University of Arizona's Department of Esports, is to enhance the Rocket League community's engagement and skill through detailed replay analysis and data visualization and provide much-needed stats for us to use. As a paid project, it also serves as a practical learning experience to refine my programming abilities, emphasizing the application of React and TypeScript in developing user-centric solutions. This project not only contributes to the esports community by providing valuable insights and fostering player development but also advances my expertise as a programmer, aligning with the University's commitment to diverse innovative education and esports excellence. It also directly benefits the University of Arizona’s Esports Program by allowing us to pull end-of-year stats to celebrate our player’s achievements and keep our casters up to date on the latest information.

Key Features

---

1. Replay Upload and Parsing

- Allow users to upload Rocket League replay files.
- Automatically parse the replay files to extract data such as scores, player statistics, match duration, and more.

2. Data Visualization

- Display interactive charts and graphs for various statistics (e.g., goals, saves, shots, boost usage).
- Use heatmaps to show player movement and actions on the field throughout the match.

3. Player Performance Analysis

- Detailed performance analysis for each player in a match, including metrics like accuracy, average speed, boost management, and aerials.
- Compare player stats across different matches to track improvement over time.

4. Match Highlights and Key Moments

- Automatically identify and highlight key moments in a match, such as goals, saves, and demolitions.
- Offer video playback of these moments if possible, or detailed timestamps and descriptions.

5. Team Analysis

- Provide insights into team dynamics, such as positioning, rotation efficiency, and pass connections.
- Evaluate team performance trends over multiple games.

6. Advanced Statistical Metrics

- Include advanced metrics like expected goals (xG), possession time, and pressure index to give deeper insights into gameplay.
- Offer breakdowns of offensive vs. defensive actions and efficiency.

7. User Accounts and Match History

- Allow users to create accounts to save and track their match history over time.
- Enable users to tag and categorize replays for easier access and organization.

8. API Integration (low priority)

- Consider integrating with external APIs for additional data enrichment, such as player rankings, tournament results, or even live match data if available.

9. Mobile Responsiveness

- Ensure the website is fully responsive and accessible on various devices, including smartphones and tablets, for users to view their data on the go.

10. Customization and Preferences

- Allow users to customize their dashboard and what statistics are most prominently displayed according to their preferences.

Key UI Elements

General Layout and Structure

- Dark Theme: A clean, dark background with vibrant accent colors for contrast.
- Navigation Bar: A top navigation bar for easy access to Home, Analysis, Community, and Player Profiles.
- Responsive Design: Ensure the website is fully responsive across devices.

Home Page

- Welcome Banner: A prominent banner with a brief introduction to "Rockety" and its purpose.
- Quick Start Guide: A section that guides new users on how to get started with uploading and analyzing their replays.

Replay Upload Section

- Upload Interface: A simple, drag-and-drop interface or file selection dialog for uploading Rocket League replay files.
- Upload Queue: Display of pending and completed uploads with progress indicators.

Analysis Dashboard

- Interactive Charts and Graphs: Visualization of game data, such as player performance, team statistics, and match outcomes.
- Match Timeline: A detailed view of key moments in the replay, like goals, saves, and demolitions.
- Player Comparison: Tools for comparing statistics between players within a match or over time.

Player Profiles

- Personal Dashboards: Customizable dashboards for users to track their own or others' performance over time.

- Achievement Badges??: Visual representation of user milestones and achievements within the Rocket League community.

Additional Features

- Search Functionality: Ability to search for players, teams, or specific matches.
- Notifications: System for alerting users about updates to their uploads, community replies, or new features.
- Accessibility Options: Features to enhance accessibility, including text size adjustments and color contrast options??

Technical Considerations

- Security Measures: Ensure user data and uploaded files are securely handled and stored.
- Loading States and Error Handling: Design for loading indicators and informative error messages for a smooth user experience.

Tech Stack

Frontend

- React: A JavaScript library for building user interfaces. Should work well for this project due to its modular nature.
- TypeScript: An open-source language that builds on JavaScript by adding static type definitions. Way better than trying to use JavaScript.
- Chart.js: Libraries for creating dynamic and interactive data visualizations in the web browser. This library will be the meat and potatoes of this website.
- Styled Components: For styling, this should work well.

Backend

- Node.js with Express: A runtime environment and a framework for building fast, scalable network applications. This is a standard for the RESTful API that this project will be building out.
- MongoDB: A NoSQL database for storing user data, replay files, and parsed analytics. Its flexible schema is suitable for the variable and complex data generated by game replays.

Data Processing and Storage

- Python: For parsing replay files and analyzing game data. I already have a bit of parsing code written in this language. Can use pandas matplot lib or Seaborn.
- RabbitMQ or Redis: As a message broker for handling asynchronous tasks, such as parsing replay files. These tools can help decouple the data processing workload from the main application flow, improving performance and scalability. I'm unsure which one to use yet as I am fairly new to using message brokers.

Infrastructure and DevOps

- Docker: For containerization, ensuring consistency across development, testing, and production environments. Standard stuffs.
- AWS or Google Cloud Platform: Cloud services for hosting, storage, and scalable computing resources. I'll most likely end up going with AWS.

- GitHub Actions: For continuous integration and deployment, automating the testing and deployment process, thereby improving code quality and deployment efficiency. I'm decently familiar with this so I'd like to implement it.

Security and Compliance

- OAuth 2.0 and JWT: For secure authentication and authorization, ensuring that user data and replay files are protected.
- HTTPS: Ensure all data in transit is encrypted using HTTPS, protecting against man-in-the-middle attacks.