Software Requirements and Design

Document

For

Group <25>

Version 3.0

Authors:

Erik P, Noah L, Keene M

1. Overview:

We are creating a web app for F1 racing using an open-source API to display car statistics and predict optimal tire strategy based on race and track conditions. This F1 API (https://openf1.org/) also includes data from past races, and we plan to include previous race simulations which can then be compared to how the real race played out. The general idea is that the user can pause the simulation at any point and be given percentage estimations on what the best tire strategy is given this point at the race. By comparing simulated outcomes with actual race results, users gain valuable insights into strategy effectiveness, enhancing their ability to make informed decisions in future races. Built on the backend framework using Flask and utilizing PostgreSQL for efficient data management, the F1 Strategy Web App ensures reliable performance and scalability. The responsive frontend, developed with Bootstrap, offers an intuitive and user-friendly interface, making complex data easily accessible and actionable.

2. Functional Requirements:

1. User Registration (High Priority)

    Description: The system shall allow new users to create an account by providing a unique username, valid email address, and secure password.

    Rationale: Enabling users to register is essential for personalizing their experience and ensuring secure access to the application's features.

2. User Authentication (High Priority)

Description: The system shall enable registered users to log in and log out securely using their credentials.

Rationale: Secure authentication is crucial to protect user data and maintain the integrity of user sessions.

3. Password Storage (Medium Priority)

Description: The system shall provide a mechanism for users to store their password.

Rationale: Facilitates user account management.

4. View Live Race Data (High Priority)

Description: The system shall display real-time race data, including current driver positions, lap times, and race status updates.

Rationale: Delivering live race information is a core feature that attracts users interested in monitoring ongoing Formula 1 events.

5. View Past Race Data (High Priority)

Description: The system shall display past race data, including current driver positions, lap times, and race status updates.

Rationale: Delivering past race information is crucial in testing our simulation's accuracy across past events.

6. Responsive User Interface (Medium Priority)

Description: The system shall ensure that all user-facing pages are responsive and accessible across various devices and screen sizes.

Rationale: Enhances user experience by providing consistent and optimal viewing across desktops, tablets, and smartphones.

7. Secure Data Transmission (High Priority)

Description: The system shall use HTTPS to encrypt all data transmitted between the client and server.

Rationale: Protects sensitive user information from potential interception and ensures data integrity during transmission.

8. Input Validation (High Priority)

Description: The system shall validate all user inputs on both client-side and server-side to prevent invalid data entries and security vulnerabilities.

Rationale: Ensures data integrity and protects against common attacks such as SQL injection and cross-site scripting (XSS).

9. Session Management (Medium Priority)

Description: The system shall manage user sessions securely, ensuring that authenticated users maintain their session state while using the application.

Rationale: Maintains security by properly handling session lifecycles. Also maintains user accounts.

10. Error Handling and Feedback (Low Priority)

Description: The system shall provide clear and informative error messages to users in case of failed actions, such as incorrect login credentials or registration errors.

Rationale: Enhances user experience by guiding users to correct issues without confusion.

11. Data Persistence (High Priority)

Description: The system shall store user account information and live race data persistently in a PostgreSQL database.

Rationale: Ensures that critical data is retained reliably for consistent application functionality.

3. Non-functional Requirements:

1. Performance

Description: The system shall load the home page and user dashboard within 2 seconds under normal user load conditions.

Rationale: Ensures a smooth and responsive user experience, preventing user frustration and abandonment.

2. Security

Description: The system shall implement secure authentication mechanisms, encrypt sensitive data, and protect against common web vulnerabilities such as SQL injection, XSS, and CSRF attacks.

Rationale: Protects user data and maintains the integrity and trustworthiness of the application.

3. Usability

Description: The system should provide an intuitive and user-friendly interface, requiring no more than 3 clicks to access the live race data from the home page.

Rationale: Enhances user satisfaction and promotes engagement by simplifying navigation and interaction.

4. Reliability

Description: The system should achieve an uptime of 99.5% during live race events, ensuring continuous availability for users.

Rationale: High reliability is critical when users depend on real-time data during ongoing events.

5. Maintainability

Description: The system's codebase should be modular and well-documented, allowing for easy updates and feature additions in future increments.

Rationale: Facilitates efficient maintenance and scalability of the application over time.

6. Compatibility

Description: The system should be compatible with major web browsers (Chrome, Firefox, Safari, Edge) and responsive across various devices (desktops, tablets, smartphones).

Rationale: Ensures accessibility and consistent user experience across different platforms and devices.

7. Scalability

Description: The system architecture shall support future scalability to handle increased user loads and additional features in subsequent increments.

Rationale: Prepares the application for growth without requiring significant architectural changes.

8. Data Integrity

Description: The system shall ensure the accuracy and consistency of user data and live race data through validation and error-checking mechanisms.

Rationale: Reliable data is essential for providing accurate live race information and user account management.

9. Accessibility

Description: The system shall comply with WCAG 2.1 guidelines to ensure accessibility for users with disabilities.

Rationale: Promotes inclusivity and allows a broader audience to use the application effectively.

10. Backup and Recovery

Description: The system shall implement regular data backups and have a recovery plan to restore user account information and race data in case of failures.

Rationale: Protects against data loss and ensures business continuity.

4. Use Case Diagram:

- Use Case 1: User Registration

Actor: New User

Description: Allows a new user to create an account by providing necessary details.

Preconditions: User is on the registration page.

Postconditions: User account is created, and the user is logged in.

- Use Case 2: User Login

Actor: Registered User

Description: Enables a registered user to log into the system using their credentials.

Preconditions: User has a registered account.

Postconditions: User is authenticated and granted access to the dashboard.

- Use Case 3: Password Recovery

Actor: Registered User

Description: Allows users to recover or reset their password through email verification.

Preconditions: User is on the password recovery page.

Postconditions: User receives a password reset link via email and can set a new password.

- Use Case 4: View Live Race Data

Actor: Authenticated User

Description: Allows users to view real-time race information, including current positions and lap times.

Preconditions: Race is ongoing, and live data is available.

Postconditions: User views updated live race data on their dashboard.

- Use Case 5: View Past Race Data

Actor: Authenticated User

Description: Allows users to view past race information, including the positions of drivers and lap times for that given period.

Preconditions: The race has already finished and all data is available.

Postconditions: User views all past race data on their dashboard and can test against the simulation for accuracy.

- Use Case 6: Logout

Actor: Authenticated User

Description: Enables users to securely log out of their account.

Preconditions: User is logged in.

Postconditions: User session is terminated, and the user is redirected to the home page.

5. Class Diagram Components:

1. User

Attributes:

user_id: Integer

username: String

email: String

password_hash: String

created_at: DateTime

Methods:

register()

login()

logout()

reset_password()

update_profile()

2. Race

Attributes:

race_id: Integer

name: String

location: String

date: DateTime

status: String

Methods:

fetch_live_data()

update_race_status()

3. Notification

Attributes:

notification_id: Integer

user_id: Integer (Foreign Key to User)

message: String

timestamp: DateTime

read_status: Boolean

Methods:

send_notification()

mark_as_read()

Relationships:

User to Notification: One-to-Many

6. Operating Environment:

- Development Machines:

    - Windows 10/11, macOS Catalina or later, and various Linux distributions to accommodate different developers' environments.

- Client Devices:

    - Any operating system that supports modern web browsers, including Windows, macOS, Linux, iOS, and Android.

- Software Components:

    - Flask: Python-based micro web framework for backend development.

    - PostgreSQL: Relational database system for storing user data and live race information.

    - HTML, CSS3: Core technologies for building the user interface.

    - Bootstrap: CSS framework for responsive and mobile-first frontend design.

    - External F1 Race Data APIs: For fetching live race data.

    - Authentication APIs: Utilizing Flask-Login for managing user sessions and authentication.

7. Assumptions and Dependencies:

- Reliable External APIs:

    - It is assumed that the external APIs providing live F1 race data are reliable, with minimal downtime and consistent data formats.

- User Base for Increment 1:

    - The initial user base is expected to be moderate, allowing the system to operate without immediate scalability concerns.

    - Developer Expertise:

    - Team members possess adequate knowledge and experience with Python, Flask, PostgreSQL, and frontend technologies (HTML5, CSS3, JavaScript, Bootstrap).

- Internet Connectivity:

    - Users accessing the application have stable internet connections to receive real-time race updates and interact with the platform seamlessly.

- Legal Compliance:

    - The use of F1 race data complies with all relevant licensing agreements and copyright laws.

    - Dependencies:

    - F1 Race Data APIs: The application depends on external APIs to fetch live race data. Any changes in the API's availability, data formats, or access policies can impact the application's functionality.

- Third-Party Libraries and Frameworks:

    - Flask Extensions (e.g., Flask-Login, Flask-Migrate): The application relies on various Flask extensions for authentication and database migrations. Updates or deprecations of these libraries can necessitate code changes.

- Development Tools:

    - GitHub: Dependence on GitHub for version control and issue tracking. Any disruptions to GitHub services can hinder collaboration and development workflows.

- Browser Technologies:

    - CSS Standards: Reliance on the compatibility and support of modern web standards by browsers. Incompatibilities or changes in browser technologies can affect the application's frontend performance and appearance.

- Team Collaboration:

    - Communication Tools (e.g., Slack, WhatsApp): Dependence on effective communication tools for team coordination this is why we are utilizing WhatsApp and Github.