Sprint 4 Artifacts & Requirements

Overview

Sprint 4 focuses on **reward systems, hints, social sharing, performance optimization, deployment, and documentation**. The backend is **Flask + SQLite**, and the frontend uses **React (Vite) with Tailwind CSS**. Security, maintainability, and performance remain high priorities.

Functional Requirements (Sprint 4)

4.1 Implement reward badges and achievements (e.g., 7-day streak)

- **Owner:** Daniel Neugent
- **User Story:** As a user, I want to earn badges for completing streaks to stay motivated.
- **Functional Requirements:**
    - FR4.1.1: Track user streaks and milestones in SQLite.
    - FR4.1.2: Display badges dynamically on the user dashboard.
    - FR4.1.3: Create reusable React components for badges and achievement notifications.

4.2 Add hint system to reveal partial hints or full solutions

- **Owner:** Brett Balquist
- **User Story:** As a user, I want hints available for challenges to help when I'm stuck.
- **Functional Requirements:**
    - FR4.2.1: Add hint content to SQLite challenge table.
    - FR4.2.2: Create backend endpoints to serve hints upon request.
    - FR4.2.3: Display hints in React components with partial/full reveal options.

4.3 Integrate social sharing for daily results on Twitter/X and LinkedIn

- **Owner:** Tej Gumaste
- **User Story:** As a user, I want to share my daily results to social platforms to celebrate my achievements.
- **Functional Requirements:**
    - FR4.3.1: Add buttons in React frontend for sharing results.
    - FR4.3.2: Prepopulate share content with user stats and challenge info.
    - FR4.3.3: Use secure OAuth or share URLs without exposing sensitive data.

4.4 Conduct performance optimizations (lazy loading, caching, compression)

- **Owner:** Jay Patel
- **User Story:** As a user, I want fast loading times and smooth interactions for a better experience.
- **Functional Requirements:**
    - FR4.4.1: Implement lazy loading of React components.
    - FR4.4.2: Enable caching of API responses in frontend where appropriate.
    - FR4.4.3: Enable compression for backend responses using Flask middleware.

4.5 Deploy full app using Render/Vercel with HTTPS and DNS setup

- **Owner:** Arnav Jain
- **User Story:** As a user, I want a reliable, secure deployment of the application with HTTPS.
- **Functional Requirements:**
    - FR4.5.1: Configure backend deployment on Render or Vercel.
    - FR4.5.2: Deploy frontend React (Vite) app on Vercel.
    - FR4.5.3: Enable HTTPS and configure DNS for production domain.

4.6 Gather beta user feedback, document issues, iterate improvements

- **Owner:** Daniel Neugent
- **User Story:** As a developer, I want user feedback to improve usability and fix issues before full release.
- **Functional Requirements:**
    - FR4.6.1: Collect feedback via forms or surveys in the app.
    - FR4.6.2: Log and track bugs and enhancement requests.
    - FR4.6.3: Prioritize fixes and iterate improvements before production release.

4.7 Finalize UI, fix bugs, perform regression testing, prepare demo presentation

- **Owner:** Brett Balquist
- **User Story:** As a developer, I want to finalize UI and ensure the app is stable for presentation.
- **Functional Requirements:**
    - FR4.7.1: Conduct end-to-end testing of all frontend components.
    - FR4.7.2: Perform regression testing on backend endpoints and database interactions.
    - FR4.7.3: Prepare presentation/demo with working features and visual consistency.

4.8 Write documentation for features, architecture, and deployment

- **Owner:** Tej Gumaste
- **User Story:** As a developer, I want clear documentation so future developers and users can understand the system.
- **Functional Requirements:**

- FR4.8.1: Document architecture of backend (Flask/SQLite) and frontend (React/Vite).
- FR4.8.2: Include setup instructions, CI/CD workflow, and deployment notes.
- FR4.8.3: Provide usage examples and API documentation for developers.
