Team 10
Team Members: Daniel Neugent, Brett Balquist, Tej Gumaste, Jay Patel, Arnav Jain
Project Name: Leetle
Project Synopsis: We are creating a webapp where people can do daily Leetcode questions similar to Wordle.

Leetle Requirements Artifacts (Sprint 4)

1. System Overview

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

2. Functional Requirements

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.
- **FR4.1.4:** Animate new badge unlocks using Framer Motion.
- **FR4.1.5:** Store achievement history for user review.

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.
- **FR4.2.4:** Limit number of hints per user per day.

- **FR4.2.5:** Add “Reveal Solution” feature with confirmation prompt.

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.
- **FR4.3.4:** Generate dynamic Open Graph images for shared results.
- **FR4.3.5:** Track number of shares for analytics and engagement.

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.
- **FR4.4.4:** Optimize database queries for leaderboard and challenge fetches.
- **FR4.4.5:** Conduct Lighthouse performance audits and fix critical issues.

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.
- **FR4.5.4:** Set up environment variables securely for API and DB access.
- **FR4.5.5:** Implement CI/CD for automated deployment and rollback.

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.
- **FR4.6.4:** Create an internal feedback dashboard for the team.
- **FR4.6.5:** Analyze user feedback trends to guide post-release improvements.

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.
- **FR4.7.4:** Verify responsiveness and accessibility across devices.
- **FR4.7.5:** Conduct final bug triage and deploy demo-ready build.

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.
- **FR4.8.4:** Create contributor guidelines and code style conventions.
- **FR4.8.5:** Document known issues, limitations, and future enhancement plans.

4.9 Present final demo to stakeholders and evaluate project success
Owner: Arnav Jain
User Story: As a team, we want to present the finished product, demonstrate key features, and gather final feedback.
Functional Requirements:

- **FR4.9.1:** Prepare and present a live demonstration covering major functionalities.
- **FR4.9.2:** Highlight key technical achievements (Monaco Editor, Leaderboard, Reward System).

- **FR4.9.3:** Showcase performance improvements and security implementations.
- **FR4.9.4:** Collect post-presentation feedback from stakeholders.
- **FR4.9.5:** Evaluate project outcomes and document lessons learned for future sprints.