# Sprint 3 Artifacts & Requirements

## Overview

Sprint 3 focuses on **advanced UI features, code editor integration, leaderboard, difficulty tagging, and admin dashboard**. The backend continues with Flask + SQLite, while the frontend uses React (Vite) with Tailwind CSS for styling. Emphasis is on **modular, reusable components**, smooth user experience, and secure handling of user data.

---

## Functional Requirements (Sprint 3)

### 3.1 Integrate Monaco Editor for in-browser coding with syntax highlighting

- **Owner:** Daniel Neugent
- **User Story:** As a user, I want an in-browser code editor with syntax highlighting to solve coding challenges efficiently.
- **Functional Requirements:**
  - FR3.1.1: Integrate Monaco Editor into React frontend.
  - FR3.1.2: Ensure editor supports multiple languages (e.g., Python, JavaScript).
  - FR3.1.3: Enable resizing and responsive layout for different screen sizes.

### 3.2 Add leaderboard with dynamic ranking based on streaks and accuracy

- **Owner:** Brett Balquist
- **User Story:** As a user, I want to see a leaderboard ranking users by streaks and accuracy to encourage friendly competition.
- **Functional Requirements:**
  - FR3.2.1: Create Flask endpoint to fetch leaderboard data.
  - FR3.2.2: Dynamically calculate rankings based on streaks and success rate.

    - FR3.2.3: Display leaderboard in a React component with responsive design.

### 3.3 Implement difficulty tagging (Easy, Medium, Hard) for challenges

- **Owner:** Tej Gumaste
- **User Story:** As a user, I want challenges labeled by difficulty so I can choose problems appropriate for my skill level.
- **Functional Requirements:**
    - FR3.3.1: Add difficulty field to SQLite challenge table.
    - FR3.3.2: Provide backend filtering endpoints by difficulty.
    - FR3.3.3: Display difficulty tags clearly in React components.

### 3.4 Build admin dashboard for CRUD on questions and user progress

- **Owner:** Jay Patel
- **User Story:** As an admin, I want to manage challenges and track user progress to maintain the platform effectively.
- **Functional Requirements:**
    - FR3.4.1: Implement Flask endpoints for creating, updating, and deleting challenges.
    - FR3.4.2: Track user progress and streaks in SQLite.
    - FR3.4.3: Create React admin dashboard with modular, reusable components.
    - FR3.4.4: Ensure admin routes are protected with authentication and authorization.

### 3.5 Add streak tracker logic on backend and frontend

- **Owner:** Arnav Jain

- **User Story:** As a user, I want a streak tracker to see how many consecutive days I've solved problems.
- **Functional Requirements:**
  - FR3.5.1: Implement backend logic to calculate streaks based on submission timestamps.
  - FR3.5.2: Display streak count dynamically on frontend components.
  - FR3.5.3: Update streak logic in real-time after successful submissions.

### 3.6 Conduct testing of Monaco Editor integration and debugging

- **Owner:** Daniel Neugent
- **User Story:** As a developer, I want to test the Monaco Editor integration to ensure it works reliably across browsers and devices.
- **Functional Requirements:**
  - FR3.6.1: Write unit and integration tests for editor functionality.
  - FR3.6.2: Verify syntax highlighting, code execution, and resizing features.
  - FR3.6.3: Log errors and debug issues in development and staging environments.

### 3.7 Implement reusable React components for leaderboard, admin dashboard, and challenge display

- **Owner:** Brett Balquist
- **User Story:** As a developer, I want consistent, reusable components to simplify UI updates and improve maintainability.
- **Functional Requirements:**
  - FR3.7.1: Modularize components for leaderboard, challenges, and admin tools.
  - FR3.7.2: Use Tailwind CSS for styling consistency.
  - FR3.7.3: Ensure accessibility and responsive design across devices.

### 3.8 Ensure secure backend handling of admin endpoints and user data

- **Owner:** Jay Patel
- **User Story:** As a user or admin, I want sensitive actions protected to prevent unauthorized access.
- **Functional Requirements:**
  - FR3.8.1: Require JWT authentication for admin routes.
  - FR3.8.2: Validate input and sanitize data before database operations.
  - FR3.8.3: Log and monitor suspicious activity in backend.

---