Requirements Document: “Pinboard” To-Do Application

Version: 1.0

Date: September 08, 2025

Author: Andrew Shevchenko

Status: Draft

1 Introduction

This document describes the requirements for the software product **Pinboard**, a web-based To-Do List application. The primary purpose of Pinboard is to provide users with a simple, intuitive, and reliable interface to manage their daily tasks.

The software will be produced as two distinct products:

1. **Pinboard-API:** A RESTful backend service built with Java and Spring Boot. It will handle all business logic, data persistence, and authentication.

2. **Pinboard-UI:** A dynamic, single-page application (SPA) frontend built with React. It will consume the API and provide the user interface for interacting with the system.

What the software will do: Allow users to create, read, update, and delete (CRUD) to-do items. Allow users to mark items as complete or incomplete. Provide a clean and responsive user interface that works on both desktop and mobile browsers. Persist user data securely between sessions.

What the software will not do (Initial Version): It will not support real-time collaboration or sharing of task lists between users. It will not send email or push notifications for task reminders. It will not have complex project management features like sub-tasks, tags, or Gantt charts.

2 User Requirements

2.1 Software Interfaces

The Pinboard application will interact with the following external systems and libraries:

- **Backend (Pinboard-API):**
    - **Spring Boot Starter Web:** For building the REST API and handling HTTP requests.
    - **Spring Data JPA / Hibernate:** For object-relational mapping and database interaction.

    - **H2 Database (Development) / PostgreSQL (Production):** The relational database management system for persisting user and task data.
    - **Spring Security:** For handling user authentication and authorization (e.g., securing API endpoints, password encoding).
    - **JJWT (Java JWT):** For creating and validating JSON Web Tokens (JWT) for stateless user sessions.
- **Frontend (Pinboard-UI):**
    - **React:** The core library for building the user interface components.
    - **React Router:** For handling navigation and routing within the single-page application.
    - **Axios:** A promise-based HTTP client for making API calls to the Java backend.
    - **CSS-in-JS Library (e.g., Styled-Components or Emotion):** For styling components in a modular way.

2.2 User Interfaces

The user interface will consist of the following key screens and components:

1. **Login / Registration Screen:** A simple form for users to create a new account or log in with an existing username and password.
2. **Main Dashboard:**
    - A header with a welcome message and a logout button.
    - An input field with an “Add” button to create new tasks.
    - A list view of all tasks belonging to the logged-in user.
    - Each task item will be displayed with:
        - A checkbox to mark it as complete/incomplete.
        - The task text.
        - An edit button (e.g., an icon) to modify the task text.
        - A delete button (e.g., a trash icon) to remove the task.
    - Visual distinction between completed (e.g., struck-through text, lighter color) and pending tasks.
    - A “Clear Completed” button to remove all completed tasks in one action.
3. **The UI will be responsive,** ensuring the layout and elements are usable on devices with different screen sizes (mobile, tablet, desktop).

2.3 User Characteristics

The intended users of Pinboard are general consumers with basic computer literacy. No specific educational level, experience, or technical expertise is required beyond the ability to use a web browser. The application is designed for simplicity, making it accessible to a wide range of users, from students and professionals to elderly individuals looking to organize simple daily chores.

2.4 Assumptions and Dependencies

- **Assumptions:**
  - Users have access to a modern web browser (Chrome, Firefox, Safari, Edge) with JavaScript enabled.
  - Users will use the application for personal task management, not for complex collaborative projects.
  - The application will be deployed on a server with a public URL that users can access.
- **Dependencies:**
  - The functionality of the React frontend is entirely dependent on the availability and correct responses of the Java backend API.
  - The project relies on the continued support and security of all third-party libraries and frameworks listed in Section 2.1.
  - The production environment depends on the availability of a PostgreSQL database server.

3 System Requirements

3.1 Functional Requirements

ID	Requirement Description	Priority
FR1	The system shall allow a user to register for a new account by providing a unique username and a password.	High
FR2	The system shall allow a registered user to log in using their username and password.	High
FR3	The system shall allow a logged-in user to create a new to-do item by entering text.	High
FR4	The system shall display all to-do items belonging to the logged-in user.	High
FR5	The system shall allow a user to mark any of their to-do items as complete or incomplete.	High

ID	Requirement Description	Priority
FR6	The system shall allow a user to edit the text of any of their existing to-do items.	High
FR7	The system shall allow a user to delete any of their to-do items.	High
FR8	The system shall allow a user to delete all of their completed to-do items at once.	Medium
FR9	The system shall persist all changes to to-do items (create, update, delete) immediately and reflect them for the user.	High
FR10	The system shall prevent unauthorized access to user data (e.g., User A cannot see or modify User B's tasks).	High
FR11	The system shall allow a logged-in user to securely log out.	Medium

3.2 Non-Functional Requirements

3.2.1 SOFTWARE QUALITY ATTRIBUTES

- **Usability:**
  - **Importance:** Critical, as the app’s value is directly tied to how easy it is to use.
  - **Measurement:** Achieved through a simple, intuitive UI with minimal clicks required for core actions (adding, completing a task). Can be measured via user testing with tasks like “Add a new task ‘Buy milk’” and “Mark the task ‘Buy milk’ as complete.”
- **Reliability:**
  - **Importance:** High. Users must trust that their task data will not be lost.
  - **Measurement:** The backend API must have an uptime of 99.9%. Data integrity is measured by ensuring that all CRUD operations executed by the user are correctly persisted to the database without errors or data corruption.
- **Performance:**
  - **Importance:** High. The application should feel fast and responsive to user interactions.
  - **Measurement:**

        - **API Response Time:** 95% of all API endpoints should respond in under 200ms under normal load.
        - **Frontend Load Time:** The main dashboard should load and be interactive (Time to Interactive) in under 3 seconds on a standard 4G connection.
- **Security:**
    - **Importance:** High. User accounts and personal data must be protected.
    - **Measurement:**
        - All user passwords must be hashed (using bcrypt) before storage in the database.
        - All communication between the React UI and Java API must be encrypted via HTTPS.
        - API endpoints must be protected against common vulnerabilities (e.g., SQL Injection, XSS). Authentication will be validated via JWT tokens on every API request.
- **Availability:**
    - **Importance:** Medium. While not a life-critical system, frequent downtime would frustrate users.
    - **Measurement:** The application should be available to users 24/7, with a target of 99.5% uptime, excluding scheduled maintenance windows.
- **Maintainability:**
    - **Importance:** High for development. The codebase should be easy to understand, modify, and extend for future developers.
    - **Measurement:** Enforced through code reviews, adherence to Java/React best practices, a modular component structure, and comprehensive code documentation.