Execution Plan for CVWO Assignment

Yu Letian

December 27, 2024

1 Objective

The goal is to build an online web forum with the following tech stack:

- **Frontend:** React.js + TypeScript
- **Backend:** Ruby on Rails
- **Database:** SQLite

The forum must include:

- An authentication system
- Create, read, update, and delete (CRUD) operations for forum threads and comments
- A tagging system

2 User Requirements

User Stories

- As a user, I want an authentication system so that I can distinguish different threads and comments made by different people.
- As a user, I want to create, read, update, and delete forum threads and comments (the basic functionality of a web forum).
- As a user, I want a tagging system so I can find topics aligned with my interests.

To be more specific, the web forum will utilize the following HTTP methods to meet the CRUD requirements:

- **POST** for creating new threads and comments
- **GET** for reading threads and comments
- **PUT** for updating existing threads or comments
- **DELETE** for deleting threads or comments

Data includes each thread or comment's creation time, modification time, content, tagging, and author (by username). Every thread will have a unique ID (e.g., T1, T2, ...) and every comment will have a unique ID (e.g., C1, C2, ...). These IDs serve as primary keys in the database.

There should be two pages:

- A **login page**, where users authenticate themselves
- A **main forum page**, containing thread listings and styled with HTML/CSS

Table 1: Use Case

Use Case	**Details**
Primary actor	Users who want to communicate online
Pre-condition	The user has already authenticated by username
Success end condition	A window appears indicating the thread/comment is successfully created or updated
Failed end condition	A window appears indicating an error occurred

Continued on next page

Table 1: Use Case (Continued)

Trigger	Clicking the new thread/comment button
Open issue	What if two users use the same username? *Proposed fix:* add a postfix number (1, 2, . . .) to distinguish identical usernames
Main process	1. The user enters his/her username. 2. The user clicks ”New Thread” to open a new window, types in a title, content, and creates a tag for it by using the symbol ‘#‘. 3. The user searches in the search box by tag. Tags are separated with the symbol ‘#‘. 4. The user can view a thread by clicking on it. 5. The user can comment by clicking the ”Comment” button, but cannot comment on a comment.
Details	1. **(1):** Check if the user is already signed up. If so, authenticate the user. If not, add a new username to the database. 2. **(4):** Given a user-input tag, the application searches the database to see if the tag exists, then finds all threads that match the required tags.

3 Learning Approach

As a newcomer to web development, I plan to invest significant time in:

- **Fundamentals:** Understanding how servers, databases, and frontends interact. by going through The Odin Project. in reflection, this actually take longer time than anticipated.
- **Framework Proficiency:** Exploring Ruby on Rails for backend logic, and React.js + TypeScript for frontend components.
- **Practice and Examples:** Play around with the skeleton project, building smaller parts, and experimenting with CRUD functionalities.
- **Debugging & Testing:** Learning how to troubleshoot issues, write tests, and ensure code reliability.
- **Documentation:** Reading official docs and community resources to stay informed and solve problems efficiently.

4 Implementation Outline

To manage the overall development process, I will:

- Initialize the project with Rails and set up the database schema for threads and comments.
- Create user authentication mechanisms (sign-up, login, logout) with secure password handling.
- Develop the frontend in React + TypeScript, integrating with Rails through RESTful APIs.
- Add tagging logic and searching functionality for a smoother user experience.
- Continuously test functionality using simple unit tests.

final modified at January 4, 2025
by Yu Letian