**Lab 3. Discussion Board Project Plan**

Diana E Torres, Sachin Kumar, Moses S Varghese

Computer Programming, Sault College

CSD230 Advanced Web Applications

Sarmad Mohammad

Thursday, May 22nd, 2025

**Lab 3. Discussion Board Project Plan**

**Project Goal**

The goal of this project is to develop a full-stack discussion board web application that allows users to register, authenticate via JWT, and post messages to public threads with an integrated chatbot that helps them with basic tasks about finding posts or threads and like/dislike button for posts. Admin users have extended capabilities to manage content. The application must demonstrate secure authentication, responsive UI, and database integration using current industry tools.

**Team Roles and Responsibilities**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Role** | **Responsibilities** |
| **Diana** | Database / Admin Functions | MongoDB Compass setup, admin functions, user roles, like/dislike data structure, GitHub management, Postman API testing |
| **Sachin** | Backend | Spring Boot APIs, JWT authentication, thread/post endpoints, like/dislike logic, chatbot integration backend |
| **Moses** | Frontend | React UI, chatbot interface, like/dislike buttons, Bootstrap layout, Axios integration, token handling |

**System Breakdown**

**Frontend (React + Bootstrap)**

* Pages: Login, Register, ThreadList, PostView
* Components:
  + Navigation Bar
  + Modal for Thread/Post creation
  + Chatbot UI (input box + result display)
  + Like/Dislike buttons on each post
* Features:
  + Token-based protected routes
  + Real-time UI updates after likes/dislikes
  + Chatbot suggestions for matching threads/posts

**Backend (Spring Boot + JWT)**

* Controllers: AuthController, ThreadController, PostController, ChatbotController
* Services: AuthenticationService, ThreadService, PostService, ChatbotService
* Security: JwtUtil, JwtFilter, SecurityConfig with role-based access
* Features:
  + JWT token generation/validation
  + Role-based permissions for admin users
  + Handling of like/dislike counts in posts
  + Keyword matching for chatbot assistance

**Database (MongoDB with Compass)**

* Collections:
  + users: username, hashed password, role
  + threads: title, creator, createdAt
  + posts: content, threadId, likes, dislikes, timestamp
* Additions:
  + Each post document includes likes and dislikes fields as integers
  + Chatbot matches user input against title and content fields using simple keyword search

**Timeline and Milestones (10-day Schedule)**

|  |  |
| --- | --- |
| **Date** | **Milestone** |
| May 27 (Day 1) | Project kickoff, GitHub setup, initial task assignment |
| May 28 (Day 2) | Base project scaffolding for React and Spring Boot, MongoDB connection |
| May 29 (Day 3) | Login/Register UI, MongoDB schema design, routing and protected views |
| May 30 (Day 4) | Bootstrap UI integration, JWT-based authentication and backend security |
| May 31 (Day 5) | Full register/login implementation, token storage and API integration |
| June 1 (Day 6) | Thread and post CRUD functionality, implement like/dislike buttons and MongoDB tracking |
| June 2 (Day 7) | Complete like/dislike backend logic, begin chatbot feature with keyword search for posts/threads |
| June 3 (Day 8) | Admin-only delete access, finalize chatbot UI and backend integration, chatbot testing |
| June 4 (Day 9) | Postman testing, UI refinement, bug fixes, documentation review |
| June 5 (Day 10) | Final project integration, GitHub cleanup, and in-class demo/presentation |

**Risks**

* **Time Constraints**: Limited timeline requires efficient parallel development and early testing.
* **Token Handling**: Incorrect token parsing between frontend/backend may block feature access.
* **Feature Overlap**: The chatbot and like/dislike must integrate cleanly without blocking base features.
* **Merge Conflicts**: Delays in pushing or pulling from GitHub branches could create conflicts close to the deadline.

**Communication and Work Plan**

* Tools Used:
  + Microsoft Teams: Document sharing, planning, and messaging
  + WhatsApp: Daily coordination and updates
  + In-Class (TSW210): In-person code reviews and integration sessions
* Meeting Cadence:
  + Quick daily check-ins via WhatsApp
  + Integration meetings every 2–3 days in class
  + Postman-based feature testing led by Diana during Days 6–9

**Communication & Work Plan**

* Primary Tools:
  + Microsoft Teams: Task assignment and file sharing
  + WhatsApp: Real-time updates
  + In-Class: Live debugging, integration meetings in TSW210
* Meeting Frequency:
  + Quick daily stand-ups on WhatsApp
  + Major integration sessions every 2 days in class

**GitHub Collaboration**

Repository URL**:**[**https://github.com/dianadeliz/CSD230Lab2**](https://github.com/dianadeliz/CSD230Lab2)

Branch Strategy

* Feature branches:
  + branch-moses
  + branch-sachin
  + branch-diana
* Each member pushes changes to their branch and submits Pull Requests when stable

**Merge Protocol**

* All merges to main go through Pull Requests
* Test changes locally before requesting a merge

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**