Message Board Project Overview Team 3

Overview

We will build a Dockerized message board application featuring user authentication, session management, and CRUD operations for boards, threads, posts, and likes. The system will consist of an Express (or Flask) backend, a PostgreSQL (or MongoDB) database, and an optional Redis service for session storage and rate-limiting.

Core Features

- Authentication & Sessions: Login & registration pages, bcrypt (or Argon2) password hashing, session management via express-session (with Redis) or JWT + refresh tokens.
- GUI Pages: Home page listing boards; board view displaying threads; thread view showing posts and comments; optional user profile page.
- Backend & API: RESTful endpoints for auth (register, login, logout), board/thread/post CRUD, and likes/upvotes; service layer for business logic; middleware for authentication and input validation.
- Database: PostgreSQL (or MongoDB) container storing Users, Boards, Threads, Posts, Likes, and session data.
- Dockerization: Docker Compose orchestrating app, db, and redis containers with named volumes for data persistence.
- Security: Input sanitization to prevent XSS/SQL injection; role-based access control (user vs. admin); rate-limiting to prevent spam; environment variables for secrets.

Architecture & Tech Stack

- UI: React/Vue SPA or server-rendered templates (Handlebars/EJS).
- Backend: Node.js with Express or Python with Flask.
- Database: PostgreSQL (or MongoDB) container.
- Cache/Session Store: Redis (optional).
- Authentication: bcrypt (or Argon2) for hashing, express-session or JWT for sessions.
- Communication: Browser ↔ REST API ↔ Database.

Docker Setup

- app: custom Dockerfile for the backend service (ports, environment variables, volume mounts).
- db: official PostgreSQL image with named volume for data persistence.
- redis: official Redis image for session and rate-limiting storage (optional).
- docker-compose.yml: defines services, networks, and volumes.

Timeline & Milestones

- June 20, 2025: Project proposal submitted; initial planning and requirements gathering.

- June 24, 2025: Set up project repository and Docker Compose skeleton (app, database, Redis).
- June 28, 2025: Implement user authentication, registration, and session management.
- July 2, 2025: Build Board and Thread CRUD functionality and basic UI pages.
- July 8, 2025: Add Post and Comment features, including likes/upvotes, and input sanitization.
- July 12, 2025: Integrate Redis for session storage and rate-limiting; perform security hardening.
- July 16, 2025: Finalize UI styling, responsive design, and polish user profiles.
- July 20, 2025: Conduct unit and integration testing, bug fixes, and update documentation (README, API docs).
- July 24, 2025: Project completion; final deliverables submitted (demo, report, code).