San Francisco Bay University

Fremont, California

A Report

on

"MindBridge: Connecting People Through Content"

[CS 526]

(*Advanced Web Programming***)**

Submitted by:

Prelisa Dahal ()

Snehal Prakash Mule ()

Submitted to:

Professor. Pragati Dharmale

Department of Computer Science and Engineering

Submission Date: December 8, 2020**5**

TABLE OF CONTENTS

TITLE **PAGE NO.**

Chapter 1: Objective and Functionality

1.1. Project Overview

MindBridge is a web-based platform dedicated to mental health and well-being resources. This system bridges the gap between mental health content creators and individuals seeking support or information. The platform serves a dual purpose:

- **For Content Creators**: It provides a dedicated space for mental health professionals, experienced individuals, and wellness advocates to share their expertise through articles.
- **For Readers**: The search functionality allows users to easily discover relevant content tailored to their specific mental health interests or concerns.

By connecting quality mental health content with those who need it most, MindBridge aims to foster a supportive community while making reliable wellness information more accessible to everyone.

1.2. Core Functionality

User Authentication

- Account creation with personal details and profile description
- Secure login with JWT-based authentication
- Password management and session handling

Content Management

- Creating and publishing posts with title, subtitle, body, and thumbnail
- Keyword tagging for improved content discoverability
- Post editing and deleting capabilities for authors

Content Discovery

- Search functionality to discover relevant content
- View posts from searched results

1.3. Target Audience

MindBridge serves a diverse community united by their interest in mental health and personal well-being:

For Content Writers:

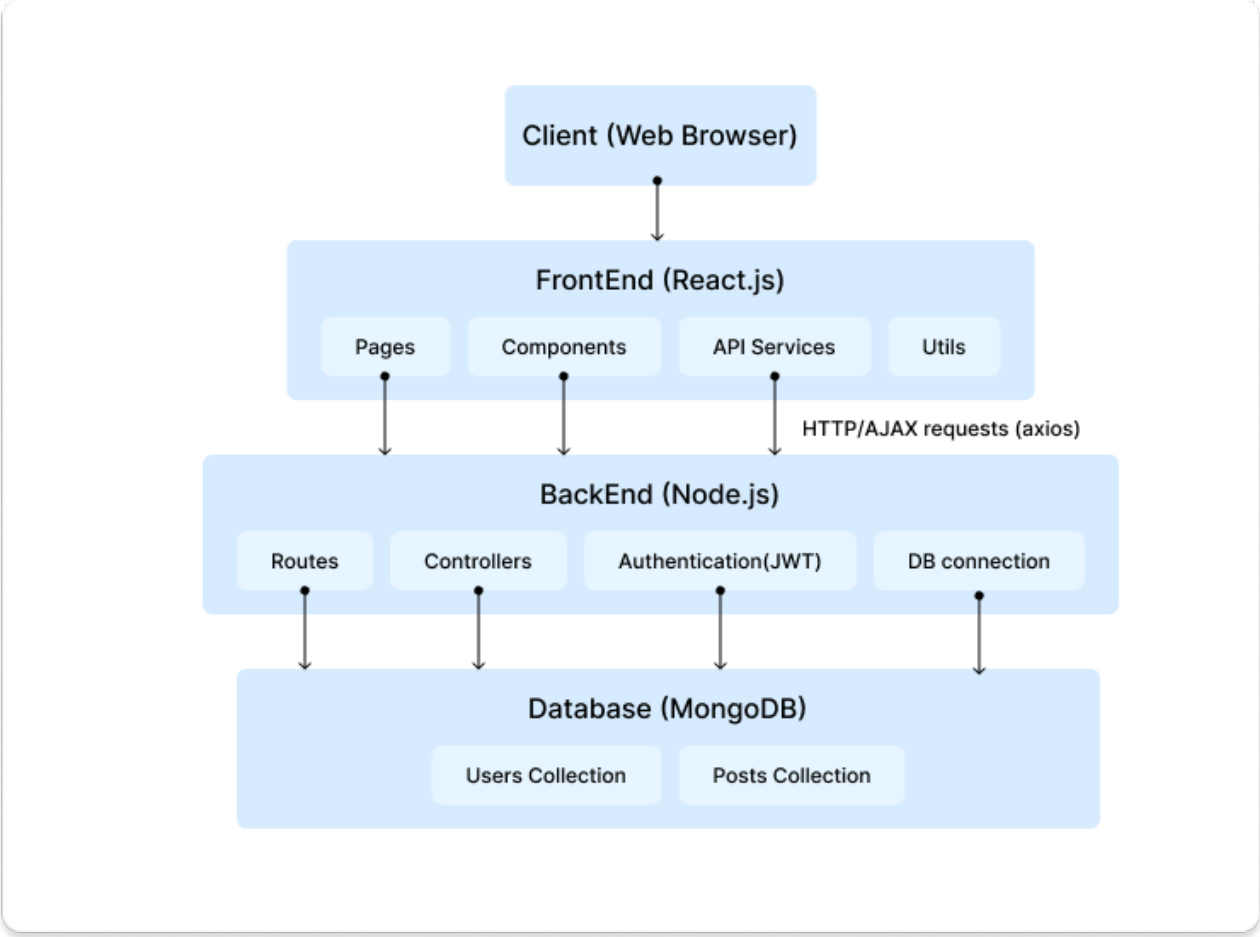
- Individuals navigating their own mental health journeys who wish to document and share their experiences
- Writers specializing in psychological well-being, emotional intelligence topics and evidence-based insights and expertise

For Content Seekers:

- People searching for reliable information about specific mental health conditions or challenges
- Individuals seeking validation and community through others' shared experiences

Chapter 2: Architecture

2.1. System Architecture Diagram



2.2. Component Architecture

Frontend (React.js)

- **Pages:** Login, Signup, Dashboard, PostEditor, SearchResult, SearchHome, PostPreview
- **Components:** Header, Footer, PostCard, SkeletonLoader
- **API integration services:** Centralized in api.js using axios for HTTP requests
- **State management:** React hooks (useState) for component-level state, localStorage for persistence

- **Routing:** React Router with protected route implementation for authenticated access

Backend (Node.js/Express)

- **Authentication controllers:** createAccount.js, login.js with JWT token generation
- **Post management controllers:** createPost.js, deletePost.js, getPosts.js, updatePost.js
- **Search controller:** searchPost.js for content discovery
- **MongoDB connection:** Mongoose ODM for database interactions
- **JWT middleware:** Token verification for protected routes

Database (MongoDB)

- **User collection:** Schema with name, email, password, description fields
- **Posts collection:** Schema with title, body, subTitle, thumbnailUrl, keywords, authorInfo
- **Relationships:** Posts linked to users via authorEmail and authorName fields

2.3. Data Flow

1. **User Registration/Login Flow**:
    - User submits credentials via frontend forms (Login.jsx/SignUpPage.jsx)
    - Data is sent to authentication endpoints (/login, /createAccount)
    - Server validates data and queries/updates MongoDB user collection
    - JWT tokens are generated with user info and 1-hour expiration
    - Frontend stores tokens in localStorage for session management
    - User state is updated (isLoggedIn=true) and redirected to Dashboard

2. **Content Creation Flow**:
    - Authenticated user creates post content using Quill rich text editor
    - Post metadata (title, subTitle, thumbnailUrl, keywords) captured in forms
    - JWT token included in request headers validates user identity
    - Backend verifies token and compares author info with token payload
    - Post data is stored in MongoDB with author information and timestamp

    - Success response triggers navigation to Dashboard with updated post list

3. **Content Discovery Flow**:
    - User searches using SearchHome or browses Dashboard posts
    - Queries are sent to /post/search or /getAllPosts endpoints
    - Server performs regex search across multiple fields (title, body, keywords)
    - MongoDB results are sorted by creation time (newest first)
    - Results are displayed as PostCard components with previews
    - User can click to view complete post content in PostPreview component

Chapter 3: Tools and Frameworks

3.1. Frontend Technologies

- **React.js**: Component-based UI library for building the interface
- **React Router**: For client-side routing between different views
- **Axios**: HTTP client for API communication
- **CSS Modules**: For component-scoped styling
- **React Bootstrap**: For UI components
- **React Icons:** For Icon components

3.2. Backend Technologies

- **Node.js**: JavaScript runtime for server-side execution
- **Express.js**: Web framework for building API endpoints
- **JSON Web Token (JWT)**: For secure authentication
- **MongoDB**: NoSQL database for storing user and post data
- **Mongoose**: For database connectivity

3.3. Development Tools

- **npm**: Package management

- **Create React App**: Frontend bootstrapping
- **Git**: Version control (inferred from project structure)

3.4. Deployment & Infrastructure

- The server is configured to run on port 8080
- MongoDB Atlas is used as the database service (indicated by connection URI)
- CORS is enabled for cross-origin requests

Chapter 4: Team Roles and Responsibilities

The project was developed by a team of two people with overlapping responsibilities:

4.1 Prelisa

- **UX/UI Design**: Created the visual design and user experience for the platform
- **Backend Development**: Implemented the Express.js server, API endpoints, and MongoDB integration
- **Frontend Development (Partial)**: Developed specific frontend components and pages
- **Integration**: Collaborated on connecting frontend components with backend services

4.2 Snehal

- **Frontend Development**: Built React components, pages, and client-side functionality
- **Backend Development:** Updating the backend for the CRUD operations
- **Integration**: Collaborated on connecting frontend components with backend services
- **Testing and quality assurance**

Both team members contributed to the overall architecture and feature implementation, with a collaborative approach to frontend-backend integration. This small team structure required versatility and cross-functional skills from both members, with you taking primary responsibility for design and backend while sharing frontend development duties.

5. Challenges, Learnings, and Future Improvements

Challenges Faced

1. **Authentication Management**: Implementing secure JWT-based authentication with proper session handling and token expiration.
2. **Rich Text Editor Integration**: Integrating and configuring the Quill.js editor to work seamlessly with content saving and retrieval.
3. **Responsive Design**: Ensuring the application works well across different devices while maintaining a consistent user experience.
4. **MongoDB Data Modeling**: Designing efficient schemas for the application's data requirements.

Key Learnings

1. **JWT Authentication Flow**: Gained deeper understanding of JWT-based authentication workflows.
2. **React Component Architecture**: Improved skills in organizing React components for maximum reusability.
3. **MongoDB/Mongoose**: Enhanced knowledge of MongoDB's document model and Mongoose's ODM capabilities.
4. **Rich Text Content Management**: Learned best practices for storing and rendering rich text content.

Future Improvements

1. **Enhanced User Profiles**: Add more customization options for user profiles, including profile pictures and social links.
2. **Search Functionality**: Implement a robust search system for finding posts by title, content, or keywords.
3. **Categories and Tags**: Add the ability to categorize posts and use tags for better content organization.

4. **Interactions and Comments**: Implement a comment system and social interactions like likes, shares, upvotes, downvotes.
5. **Image Upload Functionality**: Add direct image upload capabilities instead of using image URLs.
6. **Analytics Dashboard**: Provide authors with insights about their post performance.
7. **SEO Optimization**: Improve metadata and URL structures for better search engine visibility.
8. **Security Enhancements**: Implement additional security measures like rate limiting and enhanced password policies.

6. Conclusion

MindBridge demonstrates a solid foundation for a mental health blog platform with user authentication and post management capabilities. The project effectively implements a full-stack JavaScript application with MongoDB integration. While there are areas for improvement, particularly around security and scalability, the current implementation provides a functional platform that could be expanded into a supportive mental health content community.

The clear separation of concerns between frontend and backend, along with the modular component architecture, positions the project well for future development and feature expansion. With the recommended improvements, MindBridge could evolve into a comprehensive platform for connecting people through shared mental health experiences and well-being resources.

The collaboration between team members with complementary skills has resulted in a cohesive application that addresses an important social need. By creating a dedicated space for mental health content, MindBridge has the potential to make a meaningful impact in fostering dialogue and support around mental health topics.