Project Documentation

1 Demo Application Creation

The demo application was developed to showcase the functionalities of the API and to gather user feedback on the performance of the models. Our objectives were to create an easily accessible web application that wouldn't heavily burden the user's device. To achieve these goals, we chose to build the application using a web technology stack that our team was familiar with.

1.1 Choice of Technology Stack

Our team has extensive experience with JavaScript, making Node.js a natural choice for the backend. We aimed for a seamless development process by utilizing technologies that complement each other well. The following technologies were selected:

- **Frontend Framework**: React.js
- **Build Tool**: Vite.js
- **Styling**: Tailwind CSS
- **Database**: MongoDB Atlas
- **Backend Framework**: Node.js

1.2 Reasons for Choosing the Stack

1. **React.js**: Known for its component-based architecture, React.js allows for the creation of reusable UI components. This modularity aligns with our focus on code reusability and maintainability.
2. **Vite.js**: Vite.js offers a faster and more efficient development experience compared to traditional build tools like Webpack. Its instant hot module replacement (HMR) feature greatly enhances developer productivity.
3. **Tailwind CSS**: Tailwind CSS provides a utility-first approach to styling, enabling us to create responsive and modern UIs with minimal effort. Its integration with React.js is straightforward and enhances the overall development workflow.
4. **MongoDB Atlas**: As a fully managed cloud database service, MongoDB Atlas offers scalability, high availability, and security. It allows us to focus on application development without worrying about database management.
5. **Node.js**: Node.js is a powerful and flexible backend framework that works seamlessly with MongoDB. Our team's familiarity with JavaScript made it an ideal choice for both frontend and backend development.

1.3 Development Process

The development of the web application was led by a team member experienced with React.js. The decision to use React.js was influenced by its popularity and the availability of a large ecosystem of libraries and tools.

- **Modularity and Reusability**: The application was designed with a focus on modularity and reusability. Components were developed to be easily reused in various parts of the application.

- **Performance**: By leveraging Vite.js for the build process and React.js for the frontend, we ensured that the application was both fast and responsive.
- **Styling**: Tailwind CSS was integrated to provide a consistent and modern look and feel across the application.

1.4 Application Functionality

The demo application primarily includes two features:

1. **Home Page**: A simple landing page that provides an overview of the application and its capabilities.
2. **Show All Books**: A page that lists all the books available in the database, demonstrating the API's ability to fetch and display data.

Due to time constraints, the application does not include full CRUD (Create, Read, Update, Delete) functionalities. The current implementation focuses on demonstrating the core features of the recommendation system.

1.5 Conclusion

The demo application serves as a proof of concept, showcasing the potential of the API and the recommendation models. It provides a foundation for further development and enhancements based on user feedback and performance analysis. By choosing a familiar and efficient technology stack, we were able to develop a robust and scalable application within a short timeframe.