Abstract: AI Chat Interface – Privacy-First Web Application

Student: Nikolas Vincenti

Course: Project Java and Web Development (DLBCSPJWD01)

Phase: Finalization (Phase 3)

GitHub Repository: github.com/NikVince/ai-chat-interface

Project Overview and Motivation

The AI Chat Interface project was developed to provide a privacy-focused, user-friendly web application for real-time interaction with OpenAI's language models. The core motivation was to empower users to access advanced AI chat capabilities while retaining full control over their data. By requiring users to supply their own API keys, the application ensures that no sensitive information is stored or transmitted beyond the user's control. The solution is designed to be accessible to both general users and academic evaluators, with a simple demo mode that allows basic chat functionality without external API keys.

Technical Architecture and Implementation

The application is structured as a modern web solution, with a React.js frontend and a Node.js/Express backend. The frontend leverages Vite for rapid development and Tailwind CSS for a mobile-first, responsive design. All components are implemented as functional React components using hooks, ensuring maintainability and adherence to best practices. The main interface manages chat state, settings, and UI responsiveness, while supporting markdown rendering, error boundaries, and loading states for a seamless user experience. On the backend, Express.js provides a secure RESTful API, with middleware for CORS, security headers, rate limiting, and comprehensive input validation. The backend exposes endpoints for chat proxying, API key validation, demo responses, and model listing. At no point are API keys exposed to the frontend or stored on the server, with all sensitive operations handled securely in session and never persisted. A critical feature is the demo mode, which simulates basic AI responses for both models, including a separate demo usage cost counter. This mode ensures that the application can be partially evaluated without requiring any external credentials. In Phase 3, the architecture was extended to include a chat export feature, allowing users to conveniently download their conversation history directly from the interface.

Development Journey and Key Decisions

The project began with a clear focus on privacy, usability, and compliance with academic guidelines. The initial concept emphasized strict separation between frontend and backend, with a backend proxy pattern to protect user credentials. As development progressed, the frontend was organized into modular components for the chat interface, message display, input handling, and settings management. Custom hooks and utility functions were created to manage state and API

communication efficiently. Styling was handled exclusively with Tailwind CSS, ensuring a consistent and responsive design across devices. The backend was built with modular routes and middleware, prioritizing security and error handling. Demo mode was implemented early on in the development cycle. Throughout development, the project checklist and assignment requirements were rigorously followed. Each feature was tested for responsiveness, error handling, and documentation quality before being marked complete. The README and in-code comments were updated continuously, and architectural decisions were documented in the project's documentation directory. Several improvements were made in response to testing and feedback. Input validation and error handling were strengthened to improve security and user experience. The user interface was refined for better accessibility and mobile usability. Responding to feedback regarding the dynamism of the settings menu, the development process in Phase 3 included the implementation of a chat export option, which required both frontend and backend adjustments to support secure and user-friendly exporting of chat transcripts.

Lessons Learned

Developing the AI Chat Interface reinforced the importance of privacy by design and the value of iterative, checklist-driven development. Maintaining strict separation of concerns and never exposing sensitive data required careful architectural planning and disciplined coding practices. Frequent testing and documentation ensured that the application remained stable and maintainable throughout the project. Building a simple demo mode improved the reliability and testability of the application as a whole. The addition of chat exporting, prompted by feedback on the settings menu, underscored the importance of iterative development and responsiveness to user and evaluator suggestions.

Conclusion

The AI Chat Interface project delivers a secure, privacy-first web application that meets academic requirements for functionality, documentation, and usability. The development process emphasized security, responsive design, and maintainability, resulting in a robust solution suitable for both real-world use and academic evaluation. The project's architecture and code quality reflect a professional approach to modern web development, and the lessons learned will inform future projects in both academic and professional contexts.
