

**HCI**

**Final Assignment Report**

**AI-based virtual companion -- EmoPal**

Group Member 2252752 Liu Jiye

2352971 Xue Haotian

College Computer Science and Technology

Major Software Engineering

Instructor Shen Ying

June 10, 2025

1. **Brief Description of the Program**

**1.1 Overall Overview**

EmoPal is an AI-powered web application that serves as a comprehensive virtual emotional companion. More than a simple chatbot, it is a multi-modal interaction platform that enables users to engage in rich, emotionally aware conversations through both text and speech. Designed with the goal of enhancing emotional well-being, EmoPal specifically caters to users who may feel lonely, stressed, or emotionally overwhelmed, offering them real-time empathetic responses and companionship that feels personal and sincere.

What makes EmoPal particularly innovative is its integration of several advanced technologies into a cohesive emotional AI experience. By combining real-time sentiment analysis, voice recognition, text-to-speech synthesis, and an animated 3D avatar, EmoPal offers a user experience that closely mimics natural human interaction. It is not merely reactive; it is contextually adaptive, dynamically shaping its responses based on both the semantic content and emotional tone of user inputs.

For instance, when a user expresses sadness, EmoPal identifies the emotional undertone using Baidu NLP APIs and responds with an encouraging, supportive message generated through the DeepSeek API. At the same time, the 3D virtual avatar mirrors these interactions visually—perhaps displaying a soft expression or slower body movements to match the mood—thus reinforcing the empathetic nature of the conversation. The inclusion of speech synthesis allows the bot to "speak" in a natural tone, which can further reduce emotional distance and foster a more humanized interaction.

EmoPal’s ability to operate in both text and voice modes simultaneously opens the door to broader accessibility. People with visual impairments, reading difficulties, or even just a preference for voice interaction can benefit from its features. Furthermore, the modular nature of its design means that new capabilities—such as multilingual support, facial emotion recognition, or personalized long-term emotional profiling—can be added without disrupting the existing system.

In essence, EmoPal represents a forward-looking application of emotional computing. It blends artificial intelligence, affective computing, and user-centric design to create a system that is not just smart, but emotionally intelligent. It is a companion that listens, understands, and responds in ways that feel genuinely considerate.

**1.2 Key Technologies Used:**

**Frontend Framework**: Built using React and TypeScript, which offers component-based UI development and strict typing for more reliable and scalable code.

**Build Tool**: Vite is used for its lightning-fast development server and optimized builds.

**Speech Processing**: The Web Speech API provides both voice input (speech recognition) and voice output (speech synthesis).

**3D Interaction**: Three.js is used to render and animate a virtual character that reacts to the emotional tone of conversations.

**NLP and AI Integration**: Baidu NLP API is used for emotion and sentiment analysis. DeepSeek API generates personalized, empathetic responses.

**Serverless Backend**: A set of Node.js-based APIs run in a serverless environment, handling emotion analysis, chat processing, and auxiliary services like weather fetching.

**1.3 Codebase Structure and Modules:**

**Root Entry (main.tsx):** The application bootstraps from main.tsx, where ReactDOM renders the main <App /> component within the root DOM node.

**App Shell (App.tsx):** This is the top-level component that imports and renders the three key submodules:VirtualAgent (3D interactive character), ChatInterface (chat UI with input/output), MoodDiary (emotion tracking interface).

**Component Modules (components/)**:

**1)ChatInterface**: Includes JSX and TypeScript logic for real-time chat interaction. This component is responsible for managing user messages, invoking backend APIs, and displaying chat history. It also integrates speech recognition and synthesis via custom hooks.

**2)VirtualAgent**: Implements a 3D canvas using Three.js where a virtual avatar reflects user emotions. For example, the avatar may show concern if the user is sad.

**3)MoodDiary**: Provides a form or interface where users can record their mood over time, serving as an emotion journal.

**Hooks (hooks/)**:

**1)useWebSpeechRecognition**: Manages speech-to-text conversion using the browser’s Web Speech API. It returns status flags and recognized text.

**2)useWebSpeechSynthesis**: Handles text-to-speech conversion, allowing the AI to respond with a voice.

**Service Layer (services/apiClient.ts)**: This module abstracts the HTTP calls made to the backend APIs. It simplifies front-end logic and ensures that API interactions remain decoupled from UI code.

**Backend API (api/)**: The serverless backend includes:

**1)analyze-emotion.js**: Calls Baidu NLP APIs to detect sentiment and dialogue emotion.

**2)analyze-and-chat.js**: Orchestrates both emotion analysis and AI response generation using DeepSeek.

**3)get-weather.js**: Fetches current weather conditions, possibly used to make replies more human and relevant.

**4)\_middleware.js**: Handles request pre-processing, such as setting headers or validating input formats.

**Type Definitions (types/)**: Custom TypeScript declaration files for modules like Three.js, ensuring proper typing during development.

**Assets and Styles**: The project also includes .svg logos, .css files for styling individual components, and configuration files like vite-env.d.ts for managing environment settings.

**2.****Implemented Requirements**

EmoPal successfully meets a wide range of requirements necessary for an emotionally responsive, intelligent virtual companion:

**Natural Language Understanding**: Through the Baidu NLP API, the program can detect whether user input is emotionally positive, neutral, or negative. It can further classify specific emotions such as sadness, anxiety, or happiness.

**Empathetic Conversation**: The DeepSeek API is used to craft context-aware and emotionally aligned responses. For instance, if the emotion analysis reveals user sadness, the AI will generate replies that are supportive and comforting.

**Speech Interaction**: Users can speak directly to EmoPal, with their speech recognized in real-time and converted to text using the Web Speech API. Conversely, the AI can vocalize its replies using speech synthesis, creating a natural dialogue flow.

**3D Virtual Agent**: A central visual feature, the Three.js-based avatar provides animated feedback based on user emotion or AI sentiment, thereby enhancing immersion and user engagement.

**Emotion Logging**: Users can optionally track their emotional state over time via the MoodDiary component. This could support future features like emotion history visualization.

**Modular Architecture**: The entire application is designed in a modular fashion, where UI, logic, and API layers are decoupled. This improves maintainability and scalability.

**Environment Management**: All sensitive credentials like API keys are managed through environment variables (via .env.local), ensuring secure deployment practices.

**Responsive UI**: Styling is applied through modular CSS, and the application is compatible with modern browsers, supporting both desktop and mobile experiences.

**3.Brief Description of the Program**

**3.1 Advantages**

**Innovative Multi-modal Interaction Experience:**  
EmoPal's greatest strength is its successful integration of three interaction modalities: text, speech, and visuals. Users can not only type but also converse directly with the application, while seeing a 3D virtual avatar provide real-time feedback based on the conversation's emotional tone. This multi-channel sensory input significantly enhances the interaction's immersion and realism, making it more than a conventional chatbot and more like a lifelike 'companion'.

**Advanced and Well-Defined Tech Stack:**  
The project utilizes a modern frontend stack (React, TypeScript, Vite), ensuring development efficiency and code robustness. On the AI front, the project cleverly combines the Baidu NLP and DeepSeek APIs. The former specializes in precise, vertical sentiment analysis, while the latter is responsible for generating creative and empathetic long-form conversations. This division of labor leverages the strengths of each platform, achieving a best-practice model of "Specialized Analysis + Empathetic Dialogue".

**Highly Humanized and Emotionally-Aware Design:**  
From the core design principle (emotional companionship) to specific implementation details—such as adjusting pitch, rate, and even adding special characters in useWebSpeechSynthesis.ts to simulate a "SpongeBob" voice—all reflect a deep consideration for the user's emotional experience. The MoodDiary feature further strengthens its positioning as an emotional wellness tool, helping users track and reflect on their emotions, extending its role from "in-the-moment companionship" to "long-term care".

**Clear, Modular, and Scalable Architecture:**  
The project's code structure is clear and adheres to the principle of separation of concerns. The directories for components, hooks, services, and the serverless api are distinctly defined. For example, encapsulating speech recognition/synthesis in custom Hooks (useWebSpeech...) and API requests in apiClient.ts decouples different parts of the codebase, making it easy to maintain and test. This modular design also lays a solid foundation for adding new features like facial emotion recognition or multi-language support in the future.

**3.2 Disadvantages**

**High Dependency on Third-Party APIs:**  
The program's core functionalities—sentiment analysis and conversation generation—are entirely dependent on the external Baidu and DeepSeek APIs. This implies:

**Service Stability Risk:** Any service disruption or network issue from either API would directly cripple core functionalities. Although the code includes a fallback mechanism for when sentiment analysis fails, the AI dialogue function itself is irreplaceable.

**Cost Issues:** API calls are typically billable. As the user base grows, API call costs could increase rapidly, posing a challenge to the project's long-term operation.

**Response Latency:** Every user input must go through a long chain of requests: Client -> Backend -> Baidu API -> Backend -> DeepSeek API -> Backend -> Client. This can lead to noticeable latency in AI responses, impacting the fluidity of real-time interaction.

**Lack of Long-Term Memory and Deep Personalization:**  
The current conversation history (chatHistory) is only passed along in the API request for a single session. The history is lost once the application is closed. The AI cannot "remember" previous conversations or user preferences. Similarly, the MoodDiary data is stored in the browser's localStorage, which is lost if the user clears their cache or switches devices. This keeps the personalization experience at a superficial level and prevents the formation of a genuine, long-term, and deep user relationship.

**Relatively Simplistic 3D Avatar Interaction:**  
Although the 3D virtual avatar is a major highlight, its interaction is primarily limited to viseme animations (lip-syncing) during speech and simple state changes based on the overall emotion. To achieve the more nuanced and rich non-verbal communication mentioned in the documentation, such as "a soft expression or slower body movements," would require a much more complex animation state machine and a finer-grained mapping between emotions and actions, which is relatively rudimentary in the current implementation.

**Data Security and Privacy Concerns:**  
Storing highly sensitive personal data like mood diary entries in localStorage is insecure in a production environment. It is easily accessible to browser extensions or XSS (Cross-Site Scripting) attacks, making it unsuitable for storing private information long-term.

**4.** **How to Improve the Program**

Based on the disadvantages listed above, future improvements can focus on the following areas:

**Introduce a Database and User Authentication for Deep Personalization:**

**Build a Backend Database:** Integrate a database (e.g., Supabase, Firebase, PostgreSQL) for persistent storage of user information.

**Add a User Authentication System:** Implement user registration and login to ensure each user's data is securely tied to their account.

**Implement Long-Term Memory:** Store user conversation history, mood diary entries, and personal preferences (e.g., nickname, important dates) in the database. At the start of each conversation, a summary of key information can be retrieved and integrated into the AI's prompt, allowing the AI to truly "remember" the user and provide a hyper-personalized companionship.

**Data Visualization:** Utilize the mood diary data stored in the database to generate emotional trend charts for users, helping them visualize their emotional changes more intuitively.

**Enhance the 3D Avatar's Interaction and Expressiveness:**

**Enrich the Animation Library and State Machine:** Design and integrate more body poses and facial expression animations (e.g., nodding, thinking, confused, happy, comforting). Build a more sophisticated animation state machine to map more specific emotion labels from Baidu NLP (like like, thankful, fearful) to corresponding animations, making the avatar's reactions more precise and vivid.

**Explore Facial Emotion Recognition:** As a cutting-edge feature, integrate frontend libraries like TensorFlow.js or face-api.js. With explicit user consent, this would allow the app to recognize the user's facial expressions via their camera, enabling the avatar to mirror or respond to their emotions in real-time, achieving a higher level of emotional resonance.

**Optimize API Dependencies and Control Costs:**

**Introduce a Caching Mechanism:** For repetitive or infrequently changing requests, a caching layer can be added on the backend to reduce direct calls to external APIs.

**Explore On-Device Models:** For relatively lightweight tasks like sentiment analysis, consider using technologies like TensorFlow.js to run a small, optimized model directly in the browser. While this may increase the initial load time, it can completely eliminate the dependency on the Baidu sentiment analysis API, reducing latency and costs.

**Implement an Intelligent API Call Strategy:** Design a strategy where, for example, very short or neutral user inputs could bypass the sentiment analysis step and directly call the dialogue API, thus saving an API call.

**Strengthen User Privacy and Data Security:**

As mentioned in the first point, moving all sensitive data from localStorage to a secure, authentication-protected backend database and encrypting data in transit is the fundamental measure to protect user privacy.

**5.** **Summary and Outlook**

Through this course project, I independently designed and fully implemented EmoPal—an AI emotional companion based on multi-modal interaction. The project's goal was to explore the application of affective computing in improving human emotional well-being. Ultimately, I successfully created a web application capable of engaging in deep emotional conversations with users through text, speech, and a 3D virtual avatar. This hands-on experience of building something from scratch was incredibly rewarding.

**On a technical level**, I gained in-depth, practical experience with the full-stack development workflow of a modern web application.

**Frontend:** I became proficient in using **React** with **TypeScript** for component-based development and experienced the highly efficient build process offered by **Vite**, which ensured the robustness and maintainability of the code.

**3D Interaction:** This was my first time systematically learning and applying the **Three.js** library. I successfully implemented the loading and rendering of a 3D virtual avatar, its viseme (lip-sync) animations, and its real-time synchronization with the application's emotional state.

**Speech Technology:** By integrating the **Web Speech API**, I mastered the browser's native speech recognition and synthesis technologies. I encapsulated these functionalities into reusable React Hooks, enabling a smooth voice-based dialogue feature.

**Backend and AI:** On the backend, I used **Vercel Edge Functions** and **Node.js** to build serverless APIs. I also learned how to orchestrate calls to multiple Large Language Model APIs, such as **Baidu NLP** and **DeepSeek**, giving the application an "intelligent brain" capable of understanding emotions and generating empathetic responses.  
As a project, this experience provided me with a more comprehensive understanding and invaluable hands-on experience across all aspects of full-stack development.

**My takeaways on the Human-Computer Interaction (HCI) level were even more profound.** Professor Shen Ying's course taught me that a good interactive system is far more than a collection of features; it is fundamentally about understanding and caring for human nature.

1. **The Power of Multi-modal Interaction:** I learned that integrating text, speech, and visual interaction is far superior to any single modality. It not only enhances the naturalness and immersion of the interaction but also provides convenience and accessibility for users with different preferences or abilities (e.g., when typing is inconvenient).
2. **The Value of Affective Computing:** This project was a concrete application of Affective Computing theory. Enabling a machine to "perceive" and "respond" to user emotions is key to bridging the gap between humans and machines and to building user trust and an emotional bond. When a user sees the virtual avatar show concern in response to their sadness, the resulting emotional resonance is something that text-only interaction cannot replicate.
3. **The Immediacy and Empathy of Feedback:** The 3D avatar's lip-syncing and the AI's empathetic replies are all forms of **immediate feedback** to the user's input. This makes the user feel clearly "heard" and "understood," encouraging them to open up—a crucial aspect of an emotional support application.

Of course, the project has its shortcomings, such as the lack of long-term memory and over-reliance on third-party APIs. These are areas for continuous improvement, as detailed in Part 4 of this report.

In conclusion, the EmoPal project has been an invaluable and comprehensive practical experience. It not only honed my full-stack development skills but also gave me a new and deeper understanding of Human-Computer Interaction design, especially in the area of emotional design.

Finally, I would like to extend my special thanks to Professor Shen Ying for her dedicated guidance and inspiration throughout this course.