HUSKY AI

AI assistance in course generation and cover letters

Team Members:

1. Anutej Poddaturi - 002922161
2. Amey Bansod - 002743117
3. Sunil Srinivas K G - 002711958

Github Link : https://github.com/connectamey/husky-ai

Youtube Link : https://www.youtube.com/watch?v=nwFuxuBZW-o

Table of Contents

1. Introduction

There are now many ways to understand a topic because data and courses are developed quickly these days. Considering that the user's required level of knowledge is not specified by a curated list. In order to address this, we developed an AI web application that responds to user prompts about the courses and subjects they wish to learn. It then creates a personalized course that includes YouTube videos for support, a concept checker, and a chatbot to help with any concepts that may have been overlooked during one of the units.

When students wish to apply for jobs and are prompted to upload a cover letter during the application process, this presents another challenge. We developed a tool that can automatically generate a unique cover letter and a linked in coffee message based on the resume and job description that are attached, saving time and effort over producing a fresh cover letter for each application. The student's application process for jobs is streamlined as a result.

Our Husky AI Platform majorly provides assistance with three parts of a students or a learners career:

- ➔ The user can curate a custom course by prompting the topic and the units they want to learn
- ➔ Go through the video lectures from the curated course and take a quiz that is curated based on the topic
- ➔ Interact with a chatbot to check with any questions you have
- ➔ Curate a custom cover letter and linkedin coffee message to interact and connect with people on various social platforms.

2. Technologies Used

We have used Javascript NEXT JS as the primary language to build our web application that assists students and users with learning new topics at their very own pace. Below are few of the technologies we utilized on creating the application

1. **Next.js:** This is the main tool we used to build our web app. With Next.js, our app helps students and users learn new things at their own speed.

2. **TypeScript:** Think of TypeScript like a helper for JavaScript. It makes sure our code is clear and has type safety, so the app runs better and is easier to fix if something goes wrong.
3. **OpenAI API (gpt-3.5 turbo model):** We used this API to understand the questions the user wants to learn and curate the chapters that students are interested in. It's like having a smart friend who can assist you learn.
4. **VertexAI API:** We used this API to create custom cover letters and a linkedin tea message that the students can use to connect and increase their network.
5. **Pinecone:** This is the Vector DB we used to process the conversations and to assist users properly with their questions .
6. **LangChain:** We used this library to process data into chunks, convert and store them in the vector DB index. It assists in understanding semantic search on the application .
7. **PlanetScale:** This is where we keep all the data for the app, like a huge, secure library. It means that all your learning progress and information are safe and can be reached really quickly whenever you need it.

We put all these technologies together to make a web app that's friendly and smart, to help you learn things in a way that fits you best.

3. Custom Designed Prompts

As we learnt from the class we used various kinds of prompting techniques to curate the output and manipulate LLM to restrict the hallucinations. We for our scenarios used Role Based Prompting Technique. Few of the scenarios we created custom prompts are below:

➔ **Chapter Info Generation:**

After the user created a list of units he wants to study the LLM used the below prompt to generate a small summary for each chapter.

"You are an AI capable of summarizing a chapter. I should not see unexpected token errors in my console when a summary is generated. Make sure to adhere to the output format. ", "summarize in 250 words or less and do not talk of the sponsors or anything unrelated to the main topic, also do not introduce what the summary is about. Strictly adhere to the format needed. Summarize so that user can grasp the key points about the video\n" +chapter.name,

Link to this Prompt :
https://github.com/connectamey/husky-ai/blob/99cbbb3676e7d6caebfba125fb61c4ae84bf0826/src/app/api/chapter/getInfo/route.ts#L44

➔ **Generation of MCQ's for Concept Check:**

We wanted to create list of MCQ's to concept check on the generated topics so to do this we curated this below prompt to create custom list of Multiple Choice Questions

"You are a helpful AI that is able to generate mcq questions and answers, the length of each answer should not be more than 15 words. Produce questions. Don't mention it is fetched from a transcript. Make sure I don't see this error - Unexpected end of JSON input. Make sure you return the output as a JSON array always. Array should have at least two questions. Make sure you don't use comma after the last key value pair.Questions should be relevant to the chapter title", new Array(5).fill(`You are to generate a random hard mcq question about ${course_title} with context of the following transcript: ${transcript} . Make sure you return the output as a JSON array always. Don't mention it is fetched from a transcript. You are generating quiz questions for a viewer learning about ${course_title}. Adhere to the format needed. Array should have at least two questions. Make sure you don't use comma after the last key value pair. each object should have five properties provided. question, answer, option1, option2, option3`),

Link to this Prompt :
https://github.com/connectamey/husky-ai/blob/99cbbb3676e7d6caebfba125fb61c4ae84bf0826/src/lib/youtube.ts#L61

➔ **Chatbot restriction:**

One the user starts going through the course he generated, when he faces an issue with a small doubt instead of going back and watch the video from beginning we used below prompt to make sure the LLM doesn't hallucinate.

content: `Explain things like you're talking to a student who is eager to learn about ${courseName}. Explain in a beginner friendly way. If the question is not related to ${courseName}, then reply: You have asked an unrelated question. Please ask questions related to ${courseName}.`,},

Link to this Prompt:
https://github.com/connectamey/husky-ai/blob/f9b37cfb17179b3eb91f53933002ddfebb35425c/src/components/Chatbot.tsx#L161C12-L162C11

Similar to the above prompting techniques we have used many customized prompts to handle various scenarios and limit bot's functionality to relevant topics.

4. Data Collection and Bot Functionality

➔ Data Sources

- Description: Our project gathers data from YouTube and user-generated content, focusing on educational videos and interactive elements. This diverse dataset includes video lectures and user interactions, which enriches the content and provides varied learning materials tailored to user requests. The bot uses these inputs to create customized courses and quizzes, respond to queries, and assist in drafting personalized professional messages.
- Justification: We chose these sources because they are rich in educational content and highly relevant to the personalized nature of our application. YouTube offers extensive video lectures that can be tailored into custom courses, while user-generated content allows for real-time learning and engagement, aligning perfectly with our goal to offer dynamic and tailored educational support. This ensures our users receive accurate, engaging, and personalized learning experiences.

➔ Bot Actions

- Capabilities: Our bot is designed to enhance the educational experience by providing customized learning paths. It generates quizzes tailored to the specific course content viewed by the user and explains complex topics in a simplified manner. The bot also assists users in creating personalized professional documents, such as cover letters and LinkedIn messages.
- Complex Interactions: The bot dynamically adjusts its interactions based on the user's learning progress and preferences. It engages in sophisticated, context-aware dialogues, adapting its responses to the user's specific needs and questions. This allows for a more interactive and responsive learning environment, ensuring that users can deepen their understanding effectively.

5. Creativity and Solution Utility

➔ Innovative Features

- Unique Aspects: Our bot creates customized learning experiences that adapt in real time using cutting-edge AI. It enhances engagement and effectiveness by creating interactive learning modules that are customized to each user's learning style and speed, in addition to curating personalized courses and quizzes.
- Comparison: Our application dynamically modifies the complexity and presentation of content based on real-time user feedback, in contrast to

standard educational platforms that frequently display static content. Our platform stands out for its capacity to adjust to specific learning demands thanks to its adaptable design, which guarantees a more effective and personalized learning experience.

➔ Problem Solving
- ◆ Addressed Issues: Our application dynamically modifies the complexity and presentation of content based on real-time user feedback, in contrast to standard educational platforms that frequently display static content.
- ◆ Value Proposition: Our technology offers a tailored learning environment that improves user engagement and increases understanding. Because it is customized to meet the various demands of our customers, learning becomes more efficient and accessible while also enhancing information retention.

➔ AI Backend Development
- ◆ LLM Integration: Our program incorporates an advanced LLM to efficiently interpret and generate learning materials customized for each user's inquiry. Because of this connectivity, the bot can easily handle complicated language queries and provide correct, contextually relevant responses.
- ◆ Vector Database Utilization: To improve our data retrieval capabilities and provide fast, precise access to instructional information, we use a vector database. This technique streamlines and improves the efficiency of the learning process by enabling the bot to recognize and retrieve the most pertinent material based on semantic similarities.

6. Evaluation and Testing Process

➔ Testing Strategies
- ◆ Methodology: We used a comprehensive combination of testing methods in our approach: end-to-end tests to verify the system's whole operation from beginning to end, integration tests to make sure components operate together seamlessly, and unit tests to verify each component independently.
- ◆ Test Cases: Examples include:
  - Unit Test: We evaluated the precision of AI responses to predefined inputs.
  - Integration Test: We tested the system's ability to accurately perform database queries and return correct data.

- End-to-End Test: We simulated realistic user interactions to verify the entire application's operational flow.

➔ Evaluation Metrics
- ◆ Performance Metrics: We evaluated reaction time, accuracy of AI responses, and total system throughput as major performance factors..
- ◆ Results: The results showed that our application is dependable and efficient, with high accuracy AI replies and robust system performance even under stressful conditions.

➔ Issue Tracking and Resolution
- ◆ Identified Issues: During testing, we encountered issues such as delayed response times during peak usage and intermittent inaccuracies in AI-generated responses.
- ◆ Resolution Strategies: These were addressed by optimizing database queries and refining the AI model, respectively, which significantly improved response times and accuracy.

7. Professionalism

➔ Code Documentation
- ◆ Descriptive Headers: At the beginning of each file and major module, providing a brief overview of the module's purpose and functionality.
- ◆ Inline Comments: For complex code blocks, inline comments explain the logic or purpose behind specific operations.
- ◆ Function Documentation: Each function is preceded by comments that describe its purpose, parameters, expected inputs, outputs, and any side effects.

➔ Licensing and Citations
- ◆ We've made sure that anyone who wants to use, copy, modify, merge, publish, distribute, sublicense, or even sell copies of our software can do so by adopting the MIT license for our project. All they have to do is make sure that any significant portions of the software still bear the original copyright and license notice.

8. Conclusion

We've worked hard to make the Husky AI app something that everyone can use to learn. It's designed to be really good at understanding the topics that you need assistance in and finding information fast and accurately. We've done our best to make sure it's a helpful tool, something that hasn't really been done like this before.

We're proud of the cool new features we've built, but we know there's always room to make things even better. So, we plan to keep improving the app by testing it with different models to increase the token size, and making it easier to use, and listening to what people have to say about it.

We would like to thank everyone who helped us out. Your ideas and suggestions made a big difference.

9. Future Development

- ➔ Integrate with Large Language Model GPT 4
- ➔ Create a Community to help users upskill while they go through learning journey
- ➔ Increase token size to increase the video limit
- ➔ Gain Feedback from users to improve functionality

10. References

➔ Strict Output function that changes output based on the context: https://github.com/tanchongmin/strictjson/tree/main