**LogDetect.AI**

**Design Document**

**Description:**

Remember all the times you missed a meeting and had to ask multiple people just to know what was decided?

What if you had a dedicated assistant who you could ask anything about the meetings you missed, and it always gave you up-to-date and accurate answers?

LogDetect.AI is exactly that.

How it works is simple:

* You ask a question
* You provide a set of call logs (ordered by date)
* LogDetect parses the entire set of call logs and presents a curated set of facts that exactly answer your question.

**Features:**

* Ask any question
* Upload multiple call logs
* Re-order call logs based on your preference
* Generate facts that answer your question!
* Facts are generated by the OpenAI API and the GPT-4 model

**Prompt Strategy:**

* A Prompt Chaining Strategy was deployed
* For the first call log, all relevant facts to the question were extracted
* For all subsequent prompts:
  + The question was provided to maintain context
  + The output of the previous prompt was appended as known facts
  + The call log was provided
  + The model was asked to update the fact list based on the question and provided context
* The prompt strategy is better depicted below:

**A diagram of a problem

Description automatically generated**

**Design Decisions:**

* Initially I experimented with providing all call-logs to the model in a single prompt. I decided against this solution for the following reasons:
  + Difficulty of maintaining context when call logs are either many in number or large
  + If the number of call logs is very large, it might breach the input token limit of gpt-4
* I decided to then use a prompt chaining technique as described above. I got decent results. Here are some of the tradeoffs I made in the design:
  + No. of API calls = No. of call logs. This means it is a more costly solution. However, I decided to optimize for accuracy and correctness, over costs for this implementation.
  + Priority on maintaining context. The method I employed does well in maintaining context across call logs, whereas in the case where all call logs are provided in a single prompt, the output quality will tend to decrease with an increase in the no. of call logs

**Test Example:**

**Question:**

**What features should be included in our new software update?**

**Call Log 1:**

**1**

**00:01:05,300 --> 00:01:33,900**

**Alex: Good morning, team. To kick things off, I believe our new software update should definitely include enhanced security features. We've had feedback about vulnerabilities that need addressing.**

**2**

**00:01:34,000 --> 00:01:42,500**

**Tina: I agree with Alex on enhancing security. I also think we should integrate AI to personalize user experiences. It could analyze user data to provide customized suggestions.**

**3**

**00:01:43,600 --> 00:01:45,100**

**Raj: Both points sound crucial. Let's add these to our update. Additionally, I suggest including support for multiple languages to expand our market reach.**

**Call Log 2:**

**1**

**00:00:50,000 --> 00:01:20,000**

**Alex: Reflecting further, I think adding a dark mode is essential. Many users prefer it, especially for late-night use.**

**2**

**00:01:21,000 --> 00:01:30,000**

**Tina: Dark mode sounds like a great addition for user comfort. Let's ensure it's fully adjustable to suit various preferences.**

**3**

**00:01:31,000 --> 00:01:32,500**

**Raj: Agreed, dark mode will be part of this update.**

**Call Log 3:**

**1**

**00:00:55,000 --> 00:01:25,000**

**Alex: On re-evaluation, should we hold off on integrating AI? It might rush things, and we risk implementing it poorly.**

**2**

**00:01:26,000 --> 00:01:35,000**

**Tina: I think you're right, Alex. Let's focus on perfecting what's essential for now and revisit AI integration later.**

**3**

**00:01:36,000 --> 00:01:37,500**

**Raj: That sounds like a prudent decision. Let's proceed without AI for now.**

**Output:**

**A screenshot of a computer software update

Description automatically generated**

**Validations:**

* You cannot submit without a question and at least one call log.
* The URL you specify must point to a .txt file
* You cannot add duplicate URLs
* You cannot submit unless all URL contents are accessible by the application
* You cannot submit unless you have saved the order in which you want the call logs processed

**Future Scope:**

The Future scope of this project is immense. Here are a few applications that this technique could apply to:

* Directly convert audio to log files for a multimodal functionality
* Real time transcript to fact generation
* Automatic Minutes of the Meeting Generator

**Acknowledgement:**

I’d like to thank the Cleric team for giving me the opportunity to work on this assignment. It was probably the most fun I had building an application and I got to learn a lot while completing the assignment. I’m excited for the team to review my submission and look forward to receiving your feedback on the same.

Best Regards,

Shikhar