**Sahil Wani**  
SUID: 466848553  
Apr 15th Report

**Summary of Previous Work**

In my previous report, I laid the groundwork for enhancing my iSchool chatbot. I examined how integrating functions can help the chatbot provide more targeted responses rather than relying solely on generic LLM output. I identified three functions that are particularly relevant to my context:

* get\_course\_description(): Retrieves a brief description for a course using metadata stored in my Pinecone vector database.
* get\_course\_credits(): Extracts and returns the maximum credits information for a given course by processing the description text.
* get\_course\_schedule(): Determines the semester or offering pattern for a course (e.g., Fall, Spring, Irregular) based on the course description.

I implemented a query dispatcher (function handler) that parses user queries and decides whether a specialized function call (for credits, schedule, or description) should be executed. In cases where no specialized function is applicable, the chatbot falls back to the default Cohere-based language generation response. Overall, this phase improved the user experience by streamlining responses and increasing the precision of course-related answers.

**Best Practices in Implementing Chatbot Functions**

While implementing the functions within my chatbot, I adhered to the following best practices:

* I developed each function (for course description, credits, and schedule) as a standalone module. This separation of concerns not only makes the code easier to understand and maintain but also simplifies debugging and future extensions.
* I ensured reliable retrieval of course metadata by integrating with the Pinecone vector database. By using the SentenceTransformer for encoding and normalizing input course names (trimming and converting to lowercase), I achieved consistency in database queries.
* Recognizing that not all queries return the desired data, I designed functions to provide clear fallback messages (e.g., "Course description not found" or "Schedule information not available"). Additionally, my query dispatcher guarantees that if none of the specialized functions apply, a default Cohere-generated response is provided so that the user always receives an answer.

**Key Challenges I Encountered**

The process of integrating function-based responses into my chatbot presented several challenges. Below are some notable obstacles and the insights I gained in addressing them:

* One of the primary challenges was ensuring that the metadata stored in Pinecone matched the expected schema. Inconsistent formatting—such as variations in course names or differences in field names—created difficulties when matching user queries. To mitigate this, I normalized query inputs and adopted a structured metadata format during the data ingestion phase.
* Users might phrase their queries in various ways, which made it challenging for the query dispatcher to determine the appropriate function to invoke. I refined the dispatcher using case-insensitive keyword checks (for terms like "credit," "schedule," or "description") to improve robustness. Nevertheless, I recognize that edge cases remain, and further natural language processing refinements may be necessary.
* Embedding very short strings—such as a course name alone—sometimes resulted in less accurate vector representations. I addressed this challenge by ensuring that course names were properly preprocessed (converted to lowercase, trimmed) and by experimenting with different embedding techniques to achieve more reliable similarity matching.