CSCI 370 Final Report

Dish 2

Harry Huang
Evan Shiveley
Camila Toro Suárez
Jacob Yates

Revised June 13, 2024

CSCI 370 Summer 2024

Mr. Caleb Bartel

Table 1: Revision history

Revision	Date	Comments
New	05/17	Created introduction, functional requirements, non-functional requirements, risks, definition of done, team profile, references, and key terms sections.
Rev – 2	05/23	Created System Architecture, updated Introduction, Risks, Definition of Done, Team Profile, and References.
Rev – 3	05/31	Updated Functional Requirements, System Architecture, Software Test and Quality.
Rev - 4	06/09	Created Result, Future Word, Project Completion Progress, and Lesson Learned. Updated Software Test and Quality, Appendix, and Definition of Done.
Rev - 5	06/13	Updated Introduction, Functional Requirements, Risks, System Architecture, Software Test, and Appendix A. Created Acknowledgments

Table of Contents

I. Introduction

Dish Wireless has tasked us with creating a chatbot for internal use with the capability to return API calls and CLI commands based on user input. The goal of this project is to deliver a tool which can be used for all of Dish's needs, but also be easily swapped out and configured for the needs of any client with a catalog of CLI command and API calls they wish to have easy search functionality for. This chatbot will be semi-intelligent, taking user input and predicting what the user is looking for within said catalog. The chatbot should also have security as a high priority, keeping a clear separation between the user, Dish's systems, and Gemini AI.

II. Functional Requirements

We created a chatbot that can process either a natural language input or a command line argument. The bot identifies the context of the user input and returns an API call or a CLI command which represents an answer according to the context of the user input. The specific functional requirements are:

- An example interface to communicate with the bot for testing purposes.
- Natural Language Processing to match keywords from user input to the mapping catalog.
- Catalog which can be easily swapped out by any user, allowing for chatbot use in any context.
- Ability to interact with Gemini or Dish catalogs based on user input.
- perform the api call if sufficient context is given by the user.

III. Non-Functional Requirements

Quality assurance is the primary non-functional requirement for this project, and the markers to measure this are:

- Efficient responses to user input
- Sensible, helpful responses in relation to input keywords and context
- Backups or user guidance in any instances where a response cannot be found
- LLD 1 week creation and scope agreement.
- Two weeks of project codification.
- User Acceptance Tests (UATs) and product code delivery
- Remembering the older responses so that the users can reference back to it

IV. Risks

Technical Risks:

- User accessing confidential information
- Chatbot returning confidential information
- Inappropriate/irrelevant answers
- Gemini accessing confidential information

Skill Risks:

- Coding chatbot to access Dish catalogs without access to catalogs ourselves
- Coding chatbot to properly contextualize user input

Technical Design Issues:

- Our client needs to approve all merge requests from the “devs” branch, which will slow down our progress.
- The main problem we have currently is that when using GitLab, our client is set as the reviewer for all merge requests. This means we cannot easily merge onto even the Developer branch to get the progress from each other. This issue remains unresolved till the end. But we got around it.

V. Definition of Done

Our finished chatbot should be able to effectively interact with the user by correctly interpreting natural language inputs and returning questions to help contextualize user wants, or API calls/CLI commands to help the user access the relevant information within Dish’s inventory database. Tests should be developed to ensure that the chatbot correctly interprets a user’s input. For example, returning a CLI command to access the number of Dish servers in a certain area upon being asked “How many servers do we have in the Denver Metro area?”. The project should be delivered in a GitLab repository created by our Dish clients.

According to our Low Level Design Document, our client has provided us with a backlog of eight user stories that are mandatory to be implemented, both from the user’s perspective and from the system’s perspective. Our definition of done is having implemented these eight user stories. An example of a user story we were given is: As a user, I want to interact with AI when the prompt starts with 'AI_namebot: <prompt to be sent to gemini>'. We were also given two additional “nice to have” user stories which we don’t have time to implement within the given time but should consider as some stretch goals or future updates.

VI. System Architecture

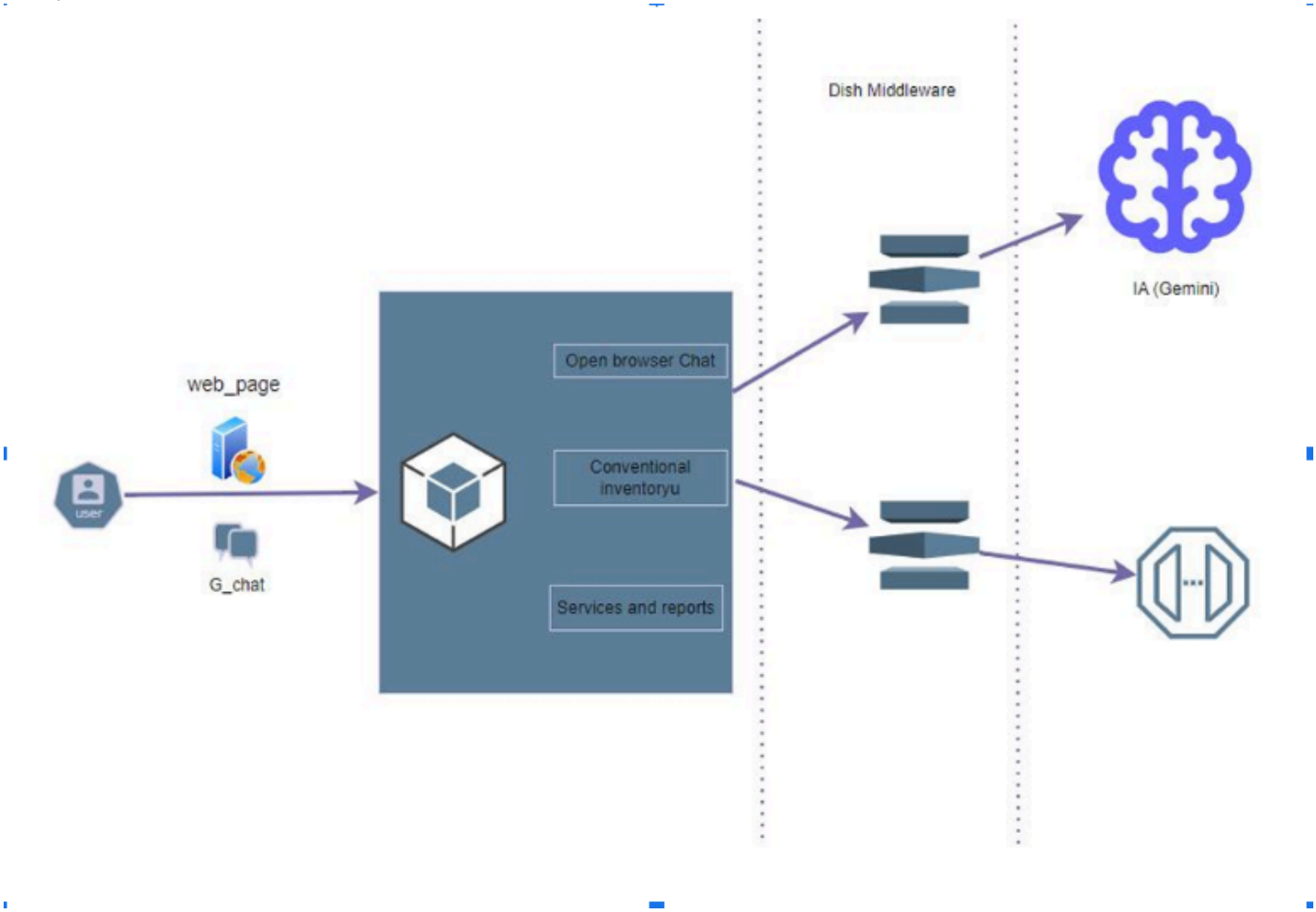


figure 1: chatbot flow

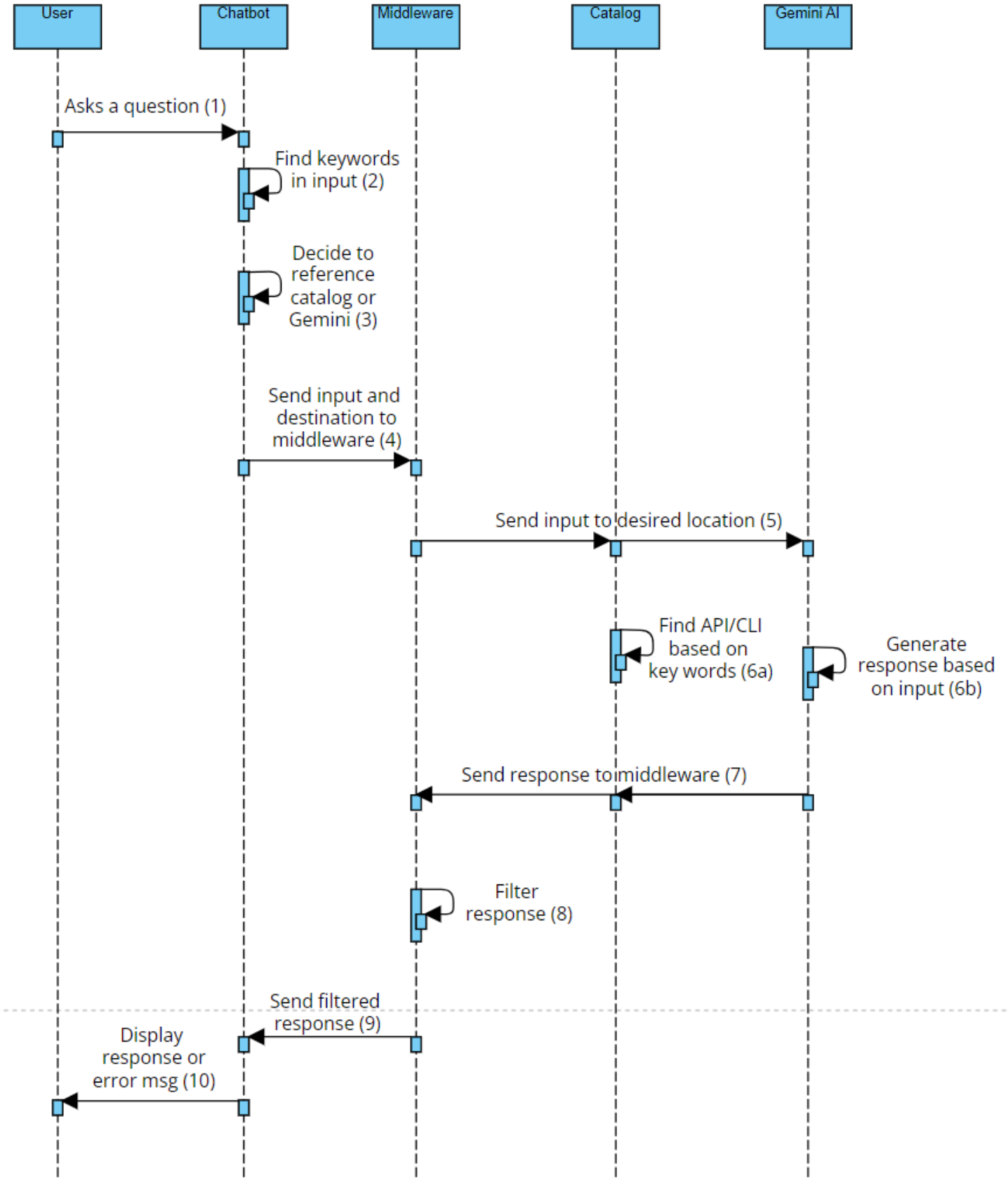
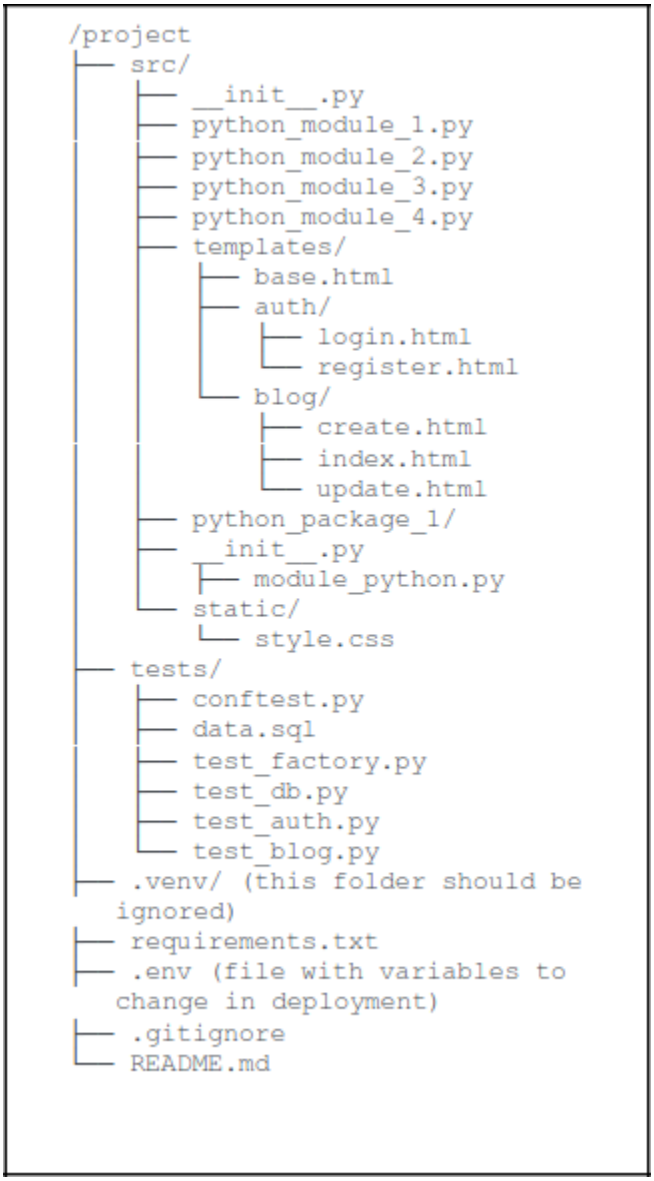


figure 2: chatbot sequence diagram



figurer 3: project structure

The three diagrams above display a high level look at the system, a sequence diagram which shows a low level design of our system, and an example of our file structure for the system. The architecture of our system will be user to chatbot input and chatbot to user output. To process this input and produce an output, the chatbot will communicate with a middleware, meaning the chatbot will never have direct access to the Catalog, and Gemini AI will never have direct access to the catalog or user input. This is for a variety of safety concerns and should provide for a secure system.

.

A complete walkthrough of the chatbot involves several steps. First, the user enters the input context. The chatbot then sends this input to the middleware, which accesses either the catalog or AI to generate an efficient response. The middleware filters and formats the response as needed. Finally, the middleware sends the API calls back to the frontend for display.

VII. Software Test and Quality

In general, one of the ways we are testing the quality of our code is through bi-weekly code reviews with our client. During our code reviews, we walk through the code we currently have and make any changes if necessary. If our client approves the changes we have made, he accepts our merge request from the pre-dev branch to the formal dev branch of our repository. The following are our test plan for our requirements:

An example interface to communicate with the bot for testing purposes

- Purpose: Allow us to easily test input / output.
- Description: A simple front-end to allow for quick and easy testing.
- Tools: Input / Output, HTML, Catalog 1 / 2
- Acceptability: Anything that visually displays input / output in an easy to read manner
- Edge Case: N/A
- Results: Successful basic front end – passed 1/1

Natural Language Processing to match keywords from user input to the mapping catalog

- Purpose: Ensure there is proper matching and that patterns are being recognized.
- Description: Test the percent match of a certain phrase and make sure the response matches one of the responses in the dictionary.
- Tools: Chatbot input/output functions and a phrase that should provoke a match.
- Acceptability: The response should match one of the keywords, it should recognize the pattern.
- Edge Case: N/A
- Results: Passed 9/9

Catalog which can be easily swapped out by any user, allowing for chatbot use in any context

- Purpose: Ensure the chatbot references the correct catalog when prompted by the user
- Description: Enter a user input that prompts the first catalog and assert that the response from the bot comes from the first catalog. The same can be tested for the second catalog.
- Tools: Assert test, chatbot input/output, catalog 1 and 2.
- Acceptability: Response will come from the corresponding catalog that it is prompted with.
- Edge Case: N/A
- Results: Passed 9/9

Ability to interact with Gemini or Dish catalogs based on user input

- Purpose: Ensure that chatbot can differentiate between user requests to Gemini vs to a catalog
- Description: Verify that chatbot accesses Dish catalogs if input starts with ‘serv_namebot:’ and accesses Gemini Ai if user input starts with ‘ai_namebot:’
- Tools: Gemini AI and a test catalog
- Acceptability: This test has generous requirements to pass - given a specific use input, we are only verifying that the chatbot accesses the correct software, not that the answer returned is correct.
- Edge Case: Adjust for typos
- Results: Passed 6/6

Perform the api call is sufficient context is given by the user

- Purpose: Ensure that chatbot returns the requested information if given enough context
- Description: Provide the chatbot with all variables needed for the call - test that the information returned correlates to the input variables
- Tools required: catalog to reference api calls and what data that api call gets at
- Threshold: no room for error - each api call has different variables and accesses a specific piece of information
- Edge Case: N/A
- Results: Passed 4/6

Efficient responses to user input

- Purpose: Ensure that chatbot can return relevant and efficient responses to the user inputs.
- Description: Provide the chatbot with valid inputs and tests if it returns relevant information.
- Tools: Assert test, chatbot input/output, catalog 1 and 2.
- Threshold: zero tolerance for error - each valid input should result in valid outputs.
- Edge Case: N/A
- Results: TBD

Sensible, helpful responses in relation to input keywords and context

- Purpose: Ensure the user will know exactly what error was made within their input and be able to correct.
- Description: Provide the chatbot input which has slight errors or lacking context. Ensure output would be useful.
- Tools: Input / Output, Catalog 1 / 2
- Acceptability: Any output which could be seen as useful to the user will be accepted.
- Edge Case: No input / Input which no keywords can be derived from.
- Results: Passed 9/9

Backups or user guidance in any instances where a response cannot be found

- Purpose: The user will understand when to rephrase their question and when the bot is confused.
- Description: The bot will provide a response to the user when the input does not match any of the patterns provided.
- Tools: Input/output, catalog 1 and 2.
- Acceptability: If the input match is less than a certain percent, the bot response should show misunderstanding.
- Edge Case: N/A
- Results: Passed 4/4

Remembering the older responses so that the users can reference back to it

- Purpose: Ensure that the users can refer back to the previously asked questions and answers
- Description: Ask the chatbot a question that depends on the previous answers - test that if the previous questions and answers were stored.
- Tools: Assert test, chatbot input/output, catalog 1 and 2.
- Acceptability: The new generated answer should be able to refer to one of the previous.
- Edge Case: Massive user inputs.
- Results: Passed 4/4

VIII. Project Ethical Considerations

The main ethical consideration we have been keeping in mind throughout the development of the chatbot has been preserving the privacy of client information. Ultimately, we are building the chatbot based on a non-Dish specific catalog in order to not expose any Dish API calls or CLI commands. The implementation of the middleware is also entirely to protect Dish privacy. We do not want the AI to directly have access to Dish information, so the bot has been built intelligent enough to choose whether to fetch a response from the Dish microservices catalog, or from the AI.

IX. Project Completion Status

In the context of our agreed upon definition of done, we have delivered a serviceable chatbot with a toggleable AI integration, the ability to differentiate between, and access, Dish microservice catalogs and Gemini AI, the ability to offer a service menu and examples of using those services, a middleware that filters information passed between the user and Gemini and vice versa, and the ability to read variables from .env files.

Specifically, our chatbot, when prompted with 'serv_namebot', properly accesses dish catalogs and identifies an API call or CLI command to generate a proper response for the users. Similarly, when prompted with 'ai_namebot', our chatbot returns a response generated by Gemini AI. We've optimized our code to make the installment of new catalogs very intuitive and modular, only requiring our clients to edit variables in one file. The chatbot also comes with a sample catalog that we've provided that offers a range of human-like responses that can help a user to maximize the bot's help. The middleware that we've implemented provides a degree of separation between the user, dish catalogs, and Gemini, to ensure the proper filtering of potentially sensitive information. We've provided an intuitive, albeit basic, frontend for the user to interact with, ensuring that Dish employees can easily ask questions without knowledge of terminal commands or how to maneuver through file structures. Lastly, our bot takes advantage of a local database to store previous user queries and the accompanying responses, granting our bot the ability to reference them in the future.

Unimplemented features involve IP logs for filtered requests and a more updated frontend. Our system could have separate logs tracking the IP address source and the user's request/message any time an AI prompt is blocked or filtered. Additionally, given more time, we would've made our chatbot installable into Google Chat, a more aesthetic and widespread software.

X. Future Work

Future work regarding our chatbot might involve new, additional catalogs that are more nuanced, and thus, require a greater degree of certainty or more niche method of deriving context to correctly match each entry to a corresponding user input. To implement a different method of NLP, additional libraries and their dependencies would likely be needed.

Another avenue for future work is an updated, more functional frontend. Currently, we only offer a text box for the user to interact with and use to make queries. In the future, though, more quality of life updates for the user like the ability to specific interactions with Gemini vs Dish catalogs without the need for 'ai_namebot' or 'serv_namebot' as a prefix before each input would make a better project. This requires more advanced HTML and JavaScript knowledge and potentially a different method of accessing the chatbot. Currently, we are using the flask app to generate an ip address that is accessible. However, with this method, the website only exists during runtime and will not be accessible after the program is shut down.

XI. Lessons Learned

This project has served to demystify the previously abstract and foreign concept of artificial intelligence and natural language processing. While many of the specific functions still exist in a 'black box', hidden behind layers of abstraction, the high level coding that Python offers has helped to break down the process of contextualizing a sentence or request from a user. For example, we've learned how to manipulate these user inputs so that NLP functions like bag_of_words() can utilize them and generate numerical data from human language.

Also through this project, our team has had to research and implement APIs so that our chatbot offers a frontend for ease of use that converses with Gemini AI and Dish catalogs. The connecting of multiple softwares for fluid interaction was something that none of us had done before, despite how prevalent it is in real world software development.

Finally, having a client and a timeline has taught us the importance of professionalism and communication. From agreeing upon the scope of the project and an MVP, to arranging bi weekly meetings for code reviews and project guidance, each of us has adopted an elevated sense of responsibility and time management. Knowing what needs to be completed each week for our client, and balancing code development with project presentations pushed us to maximize the upkeep of a backlog and instilled valuable corporate and life skills like communication, time management, and responsibility in us.

XII. Acknowledgments

The team would like to acknowledge Caleb Bartel for being a supportive and practical advisor. He was very flexible and helpful throughout.

The team would like to acknowledge Dish Wireless for being a supportive and encouraging client. The team would also acknowledge Andrew Reyes for being a super helpful and talent project leader. He had clearly communicated every piece of information needed for the project as well as provided useful help to solve any specific technical issues.

XIII. Team Profile

Camila Toro Suárez

Senior

Computer Science

Hometown: Medellín, Colombia

Work Experience: Product Development Team Intern at Maptek

Activities: Society of Hispanic Professional Engineers, Multicultural Engineering Program

I look forward to working with Dish and learning more about how AI plays a part in the chatbots we use so often!

Yunyi Huang

Senior

Computer Science

Hometown: Guangzhou, China

Work Experience: Software Engineer Intern at Modern Ark

Clubs: Esport Club, Anime Club

I am excited to be in this team and create an awesome project with them!

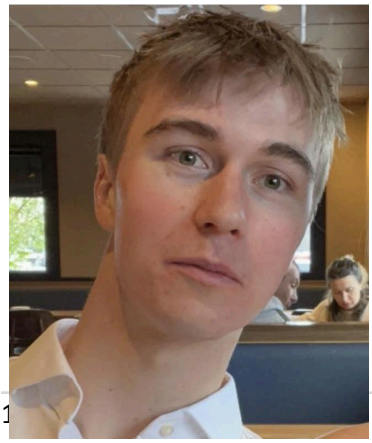
Jacob Yates

Senior

Computer Science

Hometown: Thornton, Colorado

Super excited to be on this team and to create a product for this field session!



Evan Shiveley

Senior

Computer Science

Hometown: Denver, Colorado

I'm looking forward to applying my computer science knowledge to more technical, real-world problems and working with a swag team!

References

[1]P. Engineer, “Build & Integrate your own custom chatbot to a website (Python & JavaScript),” *YouTube*. Sep. 23, 2021. Available: https://www.youtube.com/watch?v=a37BL0stIuM

[2]Pythonic Magic and Skill, “Python + Google Gemini API: Build a chat bot,” *YouTube*, Feb. 17, 2024. https://www.youtube.com/watch?v=nftMAclfeMM (accessed May 17, 2024).

Appendix A – Key Terms

Term	Definition
Catalog	*A catalog is a dictionary of API calls or CLI commands, with relevant/context words, that are used to access information in Dish*
Low Level Design (LLD)	*LLD describes the class diagrams with the methods and relations between classes and program specs*
API	*Application Programming Interface, a set of rules and protocols for building and interacting with software applications*
Command Line Interface (CLI)	*Command Line Interface, is a type of user interface that allows users to interact with a computer program or operating system by typing commands in the form of text*
Middleware	*Middleware describes software that acts as a bridge between an operating system or database and applications, especially on a network*
cURL	*A command-line tool that allows developers to transfer data to and from a server*