VCU College of Engineering

CS 25-314 Chatbot for Undergraduates Project Proposal

Prepared for

Caroline Budwell

By

Kennedy Martin

Israel Agoe-Sowah

Antony Fuentes

Eric Simoni

Date

9/26/2024

Executive Summary

Virginia Commonwealth University is committed to enhancing the student experience and ensuring that every student has access to the resources necessary to be successful. In line with this commitment, the Computer Science Department at VCU has identified a significant issue; students in the Computer Science program, both incoming and current, are facing difficulties in finding reliable information regarding their degree track. While the VCU website holds all the necessary information, its navigation is complex and often leads students to seek assistance from academic advisors and faculty for basic guidance. This reliance burdens departmental staff and leads to inefficiencies and increased workloads.

The proposed solution led by the director of the department, Caroline Budwell, is a chatbot that will be implemented on the VCU website. This chatbot will assist students in accessing degree-related information reducing their reliance on faculty for routine questions.

The main requirements are as follows: develop a chatbot that answers undergraduate Computer Science frequently asked questions and integrate the chatbot into the VCU website.

Thus far, this team has done research into the tools and techniques that could be utilized in the completion of the project and has chosen the most effective route. So far, our team has developed the front end of the chatbot and implemented Retrieval Augmented Generation (RAG). RAG allows us to pull information directly from the VCU Computer Science web pages and then leverage a LLM for word processing and conversation functionality.

Completion of this project is expected by May 2025. The timeline for the remainder of the project is as follows: Refactor the project proposal to include RAG and the current testing and environmental changes by February 23rd, propose and implement chatbot to IT by March 14th, construct the final project poster by March 25th.

Table of Contents

Section A. Problem Statement

At Virginia Commonwealth University (VCU), we place a strong emphasis on enhancing the student experience and ensuring that every student has access to the resources needed for academic success. However, it has recently come to our attention that students in the Computer Science program, both current and incoming, are facing significant challenges in efficiently obtaining accurate information about their degree and subsequent requirements.

Although the VCU website provides the necessary information, students often find it difficult to navigate. As a result, many students frequently turn to academic advisors or faculty members for assistance, creating additional workloads for staff and leading to inefficiencies. This issue is particularly concerning given the recent introduction of the Bachelor of Arts (BA) in Computer Science, which is expected to increase the number of enrolled students and further exacerbate these difficulties.

If left unaddressed, this problem will likely lead to increased faculty time dedicated to answering routine questions, incurring indirect costs for the department. Additionally, students may experience confusion regarding their academic paths, which could result in unnecessary course enrollments, delays in graduation, or even changes in major due to misinformation or misunderstanding.

This project aims to resolve these issues by streamlining the process for students to access degree-related information and reducing the reliance on faculty and advisors for basic inquiries. Leading the project is the Undergraduate Director of the Computer Science department, Caroline Budwell.

The history of chatbots began in 1966 with ELIZA, one of the first programs to simulate conversation and project an understanding of human language. ELIZA laid the foundation for how chatbots operate today. Over the years, many chatbots were developed, each utilized in different ways but all simulating human interaction.

The next big milestone occurred in the early 2000s with the uptick of machine learning. This advancement allowed chatbots to evolve from simple, template-based systems to more sophisticated models using neural networks. By the late 2010s, chatbots became more widespread as the tools and technologies needed to build them became more accessible.

Today, chatbots have reached new levels of capability thanks to transformer architectures, which allow for human-like interaction like never seen before. Chatbots are now seen in various settings and are implemented across the web.

This project focuses on implementing a chatbot on the VCU website to assist with answering general questions about curriculum and degree tracks. In the future, this project could be scaled to encompass more ambitious goals, such as dynamically creating web pages based on user queries or even acting as a teaching assistant to guide students through course material with step-by-step explanations.

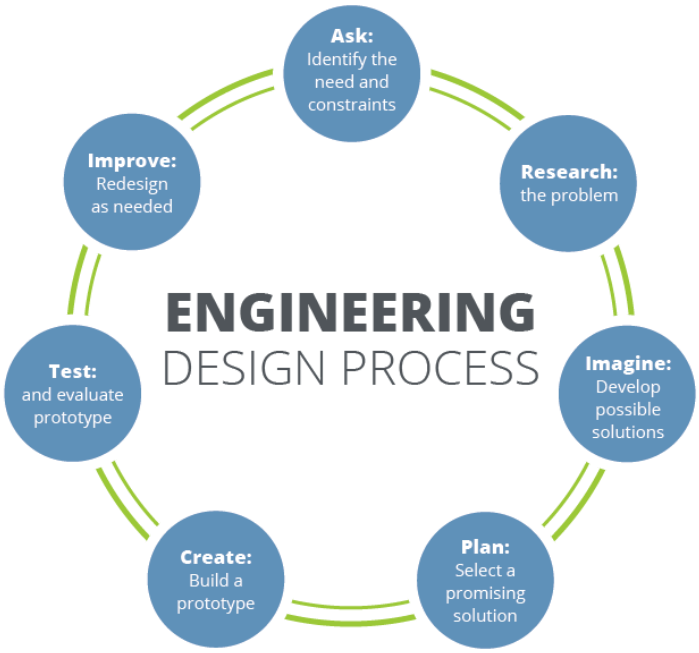


Figure 1. The iterative nature of the engineering design process [2].

Section B. Engineering Design Requirements

B.1 Project Goals (i.e. Client Needs)

Our project aims to fulfill the needs of undergraduate computer science students by providing an efficient and user-friendly chatbot that answers FAQs. Students will face challenges in accessing academic information, either because of the information being dispersed across different resources or because they must contact an advisor/administrative staff. This often leads to delays in decision-making, especially during peak times like course registration. This chatbot will address this issue by consolidating the most common inquiries into a single, accessible platform where students can reduce the time they spend searching for answers and easing the burden of advisors having to answer similar questions multiple times a semester. The chatbots 24/7 availability ensures that students have access to this information outside of staff business hours.

- To enhance accessibility by providing 24/7 access to information and reducing reliance on faculty and department office hours.
- To increase efficiency in answering common queries about course registration, prerequisites, office hours, academic requirements, etc. by automating FAQs.
- To allow students to improve their decision-making by giving access to information for course planning, which can help with avoiding registration delays and missing deadlines.
- To improve the student's engagement with the university's website, by providing it with a responsive and helpful chatbot feature.

B.2 Design Objectives

- Provide Real-Time Information:
    - The chatbot will be able to provide answers to 90% of FAQs within a reasonable time window. Which will be achieved through internal knowledge base covering of topics such as course registration, deadlines, prerequisites, etc.
    - The time between a student's question and the chatbot's response should remain under a reasonable time threshold.
    - This feature should be done and fully operational by the end of this semester.
- Handle High Traffic Efficiency:
    - The chatbot will be designed to handle a large number of concurrent users without performance degradation. This will allow the system to function well during high-traffic periods like registration week.

        - We will have to conduct tests during peak usage to ensure that the system maintains an acceptable performance level with us wanting to keep response times for all users to be under a reasonable amount of time.
- Integrating with University Systems:
    - The chatbot will integrate with the university's database to provide dynamic answers.
    - Integration success will be measured by the chatbot's ability to pull real-time data from the university's website.
    - APIs will be used to fetch this data, and integration will be tested with student queries to ensure accuracy.
- Adaptability:
    - The chatbot will be designed to allow for easy updates and the ability to expand beyond the Computer Science department.

B.3 Design Specifications and Constraints

- Response Time Constraint:
    - The chatbot must respond to the user's question within a reasonable amount of time. Given that the system will have to handle peak load scenarios of a large number of concurrent users.
    - We will have to simulate traffic conditions to verify that performance under load, ensuring response time stays under a certain amount of time.
- Data Privacy Constraint:
    - The chatbot must adhere to the university's data privacy standards.
    - The chatbot will not collect or store any personal information from users, its functionality is to answer FAQS and provide information that can be found on the VCU website.
- Budget constraint:
    - Total development cost for the chatbot should not exceed $1000. This includes expenses related to cloud hosting, APIs, and other external services that may be required for development.
    - The budget relies solely on the usage costs associated with open AI. These costs can vary depending on the model chosen for language processing, however with our implementation costs for query and response are < .01 each.
    - Costs will scale directly with usage of the chatbot, meaning during times of higher interaction, costs will increase and subsequently during times of lower interaction, costs will decrease.
    - For more information on pricing visit: Pricing | OpenAI
- Maintenance and Scalability Constraint:

    - Chatbot has to be designed for easy maintenance, allowing updates to the FAQs database or the chatbot's knowledge base without requiring a complete system overhaul.
    - Scalability: The chatbot should be able to allow future expansion into other academic departments.
- Platform Compatibility Constraint:
    - The chatbot has to be compatible with VCUs existing web infrastructure and has to be seamlessly integrated into the website. It also has to be able to function across different devices, such as desktops, phones, and tablets.
    - Compatibility tests will have to be conducted across multiple browsers and device types to ensure smooth functionality.
    - The chatbots interface will have to adapt to various screen sizes, ensuring that every user will be able to access this feature equally without any constraints.
- Error Handling and Reliability Constraint:
    - The chatbot has to have error-handling features, in which the chatbot will be able to direct the user to the appropriate department or offer an email or site in which they can reference, if the user's question couldn't be answered.
    - If the chatbot fails to retrieve a response within the given time window, the system will suggest alternate actions.
    - Reliability will be tested by intentionally creating failure scenarios to verify the system can handle these questions smoothly.

Section C. Scope of Work

C.1 Deliverables

Preliminary Design Report: Details of the chatbot's design framework, including its architecture, functional requirements, and user interface plans.

Functional Chatbot Prototype: A working chatbot that answers basic queries regarding the degree program, deployed on a website.

Final Design Report: A comprehensive report covering all aspects of the chatbot's development, functionality, and technical details.

User Manual: Create a user manual explaining how the chatbot works for both students and admins responsible for maintaining it.

Requirements.txt File: Create a requirements.txt file for easy environment setup and maintenance.

Fall Poster and Presentation: Presentation summarizing the project's progress and key milestones to date.

Capstone Poster and Presentation: Presentation showcasing the completed product, and its utility.

C.2 Milestones

Milestone	Description	Est. Time
Preliminary Design Report	Document initial designs for the chatbot.	1-2 week
Development of Computational Model	Set up the back-end logic, including NLP or rule-based systems.	3 weeks
Frontend UI Design	Build a chatbot user interface (UI) for interaction.	2 weeks
Prototype Completion	Finalize a working prototype for the chatbot.	3 weeks
Data Acquisition and Analysis	Gather Real user data from beta testing.	2-3 weeks
Create a user manual	Document how users can interact with the chat bot, and how to maintain it for admins.	1 week
Capstone EXPO Poster & Presentation	Final poster and presentation for project demonstration.	1-2 week

C.3 Resources

Software

1. Python code to implement RAG in combination with Flask for backend.
2. Flask to add support for web-based interactions.
3. HTML to create the front end web pages for users to interact with.
4. Visual Studio Code as an IDE to edit any code with.
5. Javascript to code how the web page works.

Version Control

1. Github to keep a repository of our code and the changes we make to it.

Section D. Concept Generation

Concept 1: Interactive Chatbot

- **Description**: This chatbot focuses on providing quick answers to frequently asked questions (FAQs) about the computer science program. It uses a RAG approach.
- **Pros**:
    - Straightforward implementation with minimal computational resources.
    - Easily extendable by adding new/more web pages to scrape for data.
- **Cons**:
    - Answers are limited to information available on webpages.
- **Risks**:
    - It may hallucinate, the probability of which depends on LLM.

Concept 2: Personalized Degree Planner

- **Description**: This chatbot guides users in planning their degree path by asking questions about their interests, completed courses, and career goals. It integrates with course schedules and prerequisites.
- **Pros**:
    - Highly engaging due to its personalized nature.
    - Helps students make informed decisions about their education.
    - Builds on existing university databases.
- **Cons**:
    - Requires integration with institutional systems, which may pose technical and administrative challenges.
    - Higher computational and design complexity.
- **Risks**:
    - Personalization would require tracking of users.
    - Longer development time.

Concept 3: Virtual Peer Advisor

- **Description**: This chatbot mimics a peer advisor, offering conversational support, tips for success, and recommendations for student resources like tutoring or clubs. It uses natural language processing (NLP) to make conversations feel natural.
- **Pros**:
    - Mimics human interaction, increasing student comfort.
    - Covers a broad range of topics, not just academic planning.
    - Promotes campus resources and peer interaction.
- **Cons**:
    - NLP implementation requires advanced programming and testing.
    - Misunderstanding user input could result in frustration.

- **Risks**:
    - May require ongoing training with new data.
    - Possibility of hallucinations giving students false information.

Section E. Concept Evaluation and Selection

Based on the design objectives, constraints, and client discussions, the following selection criteria are proposed for evaluating the chatbot concepts:

1. **Performance**: How effectively the chatbot fulfills the intended purpose (e.g., accuracy of responses, personalization, conversational depth).
    - **Metric**: Percentage of successful responses during testing.
2. **Ease of Implementation**: The complexity and effort required to design, develop, and deploy the chatbot.
    - **Metric**: Estimated development time.
3. **Cost**: The financial resources needed for development and maintenance.
    - **Metric**: Total estimated cost.
4. **Reliability**: The chatbot's stability, ability to function without failure, and accuracy over time.
    - **Metric**: Average correct answers.
5. **User Engagement**: How effectively the chatbot engages users and retains their attention.
    - **Metric**: Average session duration.
6. **Scalability**: Ability to grow or adapt to additional features or increased usage.
    - **Metric**: Time required to implement an additional feature
7. **Integration**: Likelihood of successful integration due to technical, user, or data-related challenges.
    - **Metric**: Likelyhood to not break already existing systems.

All metrics have been scaled on a 1 to 10 scale.

	Weighting
Performance	0.2
Ease of Implementation	0.1
Cost	0.3
Reliability	0.2
User Engagement	0.05
Scalability	0.05
Integration	0.1

Criteria	Weight	Chat Bot	Degree Planner	Virtual Advisor
performance	0.20	7 (1.4)	8 (1.6)	9 (1.8)
Ease of Implementation	0.10	9 (0.9)	6 (0.6)	5 (0.5)
Cost	0.30	9 (2.7)	7 (2.1)	6 (1.8)
Reliability	0.2	8 (1.6)	7 (1.4)	6 (1.2)
User Engagement	0.05	6 (0.3)	8 (0.4)	9 (0.45)
Integration	0.1	9 (0.9)	7 (0.7)	6 (0.6)

Total Scores

Chatbot: 7.8
Degree Planner: 6.8
Virtual Advisor: 6.35

Section F. Design Methodology

F.1 Architecture/High-level Design

There are two major components to building our chatbot: python program and the front end. Below are in depth descriptions of each area and their relation to each other.

Python Program: The Python program handles several critical tasks. First, it implements Retrieval augmented generation (RAG). RAG enables us to scrape web pages for data and then utilize a LLM for word processing and conversation functionality. This enables us to reuse the Python program for implementation across different VCU web pages/departments with minimal changes. Administrators only need to input the pages they want scraped and then add the HTML to the page where the chatbot should appear. The program scrapes the specified web pages and creates 'Document' objects. Afterward, an embedding model is used for word vectorization and other processing techniques to generate a vector-based index. When a user enters a query, it is processed using the indexed data, and a response is generated. Finally, the response is sent to the chatbot interface using Flask.

Front-End: The front-end of the application consists of HTML elements, JavaScript, Python backend handlers (FastAPI and Websocket), and CSS files for easy integration onto the VCU Computer Science website. Our chatbot integrates a FastAPI backend with JavaScript and WebSocket technology to enable real-time communication. FastAPI serves an HTML page using Jinja2 templates and manages both HTTP POST requests and WebSocket connections. A route at "/chat" handles standard message exchanges with the chatbot, while the "/ws" endpoint maintains a live WebSocket connection for instant message delivery. On the client side, JavaScript dynamically renders the chatbot UI, manages user input, and displays messages. It establishes a WebSocket connection to the backend, sends user messages, and listens for bot responses to update the chat interface in real time. The interface includes a collapsible chat window, styled with CSS, and supports both button clicks and keyboard input for message sending. This architecture provides a seamless, interactive chatbot experience directly in the browser.

F.2 Experimental/Testing Methods

Testing for this application has already begun and will continue throughout development. The main approach is trial and error, and through this process, we have already identified and started addressing several issues.

We begin testing by asking simple questions to ensure basic functionality, such as input/output handling, proper formatting, and effective communication between the front-end and back-end. Once we confirm that the application is performing these basic tasks correctly, we move on to testing for accurate and appropriate responses.

Next, we test with simple questions, verifying that the responses are correct. Then, we gradually move on to more complex queries. During this process, we encountered issues that

we are currently addressing, including issues with finding/creating the best answer if a query is not explicitly answered on the web pages.

Another test is for the frequency of hallucinations and if they can be handled. Using an LLM the chatbot can create new information that will most likely be wrong and especially if it is asked a question that it can not find in any of the information given to it. To test this we ask the bot questions that fall outside the scope of what it should know. If it can realize it doesn't have the information then it should portray that to the user.

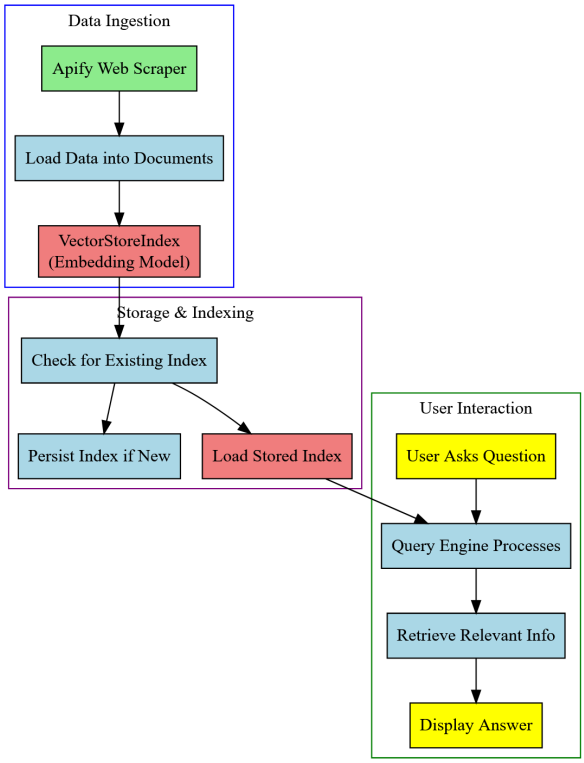
F.5 Validation Procedure

Throughout the development process, we have the advantage of working closely with our faculty advisor, allowing us to regularly validate the application's needs and make adjustments as necessary. Furthermore, we are going to utilize VCU faculty and staff to find improvements and implementation requirements.

Section G. Results and Design Details

This section describes our small-scale prototype, designed to help students access program information efficiently. The prototype utilizes retrieval augmented generation with data extracted from the college's bulletin board, ensuring up-to-date content. We are using a Natural Language Model (NLM) for processing, allowing it to generate relevant queries. Currently, we have been able to make a proof of concept prototype but we need to expand the scale and improve the general architecture of the application before it is implemented.

G.1 Prototype Diagram:



G.2 Prototype details:

- Subscription Details:
  - Utilizing RAG means we need an Apify and OpenAI key, both of which require emails for their subscription management. The emails should be listed under an administrator for each instance of the chatbot.
- **User Interface Integration**:
  - **HTML Implementation**: Embed chatbot container into VCU pages.
  - **Functionality**:
    - User inputs are sent to the backend for website queries.
    - Chatbot responses are displayed within the container.
  - **Code**:
    - **JavaScript**: Fetches user input and updates the chatbot container with JSON-formatted responses from the backend.
    - **CSS**: Styles the chatbot for an enhanced user experience.
- **Backend Logic**:
  - **Technology Used**: Python.
  - **Functionality**:
    - Processes user queries using NLP to extract keywords.
    - Uses extracted keywords to query the website for relevant information.

- Formats the retrieved data into a conversational response.

G.2 Current Testing Results:

During our testing process the main issue is when there isn't enough relevant information found. This causes the chatbot to try and create an answer that may or may not be suitable for the query asked. To fix this we may need to implement prompt engineering in order to format responses to be used when the correct information cannot be found from the sites that are referenced. This brings up another issue, there needs to be a way to maintain a report of queries that cannot be answered so that the information can either be added to the website and the chatbot has the ability to answer it. This might require implementing some sort of dashboard where that information can be held and flagged for administrators to fix.

Section H. Impacts of Design - VCU

Expanded Resource Allocation for Students

The integration of this chatbot technology at Virginia Commonwealth University (VCU) aims to alleviate inefficiencies in resource allocation for both students and staff. This tool provides immediate assistance for tasks such as class registration, financial aid inquiries, and general information dissemination. By automating routine tasks, the chatbot enables:

1. **Streamlined Administration**: Staff and faculty can focus on high-priority tasks that require personalized attention, such as academic advising or resolving complex administrative issues.
2. **Cost Savings**: Automation reduces the operational costs associated with manually handling frequently asked questions or repetitive administrative tasks.
3. **Enhanced Student Experience**: Students gain access to a consistent and reliable support system available 24/7, mitigating stress during high-pressure periods like registration.

Privacy Impact

Ensuring the privacy of chatbot users is fundamental. The following measures address privacy concerns:

1. **No Data Storage by Default**: The chatbot is designed to operate without retaining personal identifiable information (PII) unless explicitly necessary for functionality or user consent is provided.

2. **Secure Data Handling**: In scenarios requiring data retention, all information will be stored on a secure, VCU-approved server with limited and controlled access. This ensures compliance with data protection standards.
3. **Privacy-By-Design Approach**: Privacy considerations were integral to the system's design, adhering to ethical and legal obligations to protect users.

Section I. Cost Analysis

In developing the VCU Computer Science chatbot, we evaluated several platforms to determine the most cost-effective and functional solution. Here's an overview of each platform considered, including their features and associated costs:

1. Microsoft Bot Framework: This comprehensive platform facilitates the creation and deployment of conversational AI solutions. While the framework itself is free, hosting on Microsoft Azure incurs costs ranging from $5 to $10 per month, depending on usage and traffic. This results in an estimated annual cost between $60 and $120.

2. Vercel: Optimized for deploying static and serverless applications, Vercel offers simplicity and speed. However, it lacks built-in chatbot tools, necessitating integration with other frameworks. Advanced hosting plans start at $20 per month, leading to an annual cost of approximately $240.

3. ChatterBot: An open-source Python library, ChatterBot enables the creation of rule-based chatbots. While the library is free, hosting costs depend on the chosen platform. For instance, hosting on Google Cloud Platform would cost approximately $5 to $20 per month, resulting in an annual expense between $60 and $240.

4. Dialogflow: Developed by Google, Dialogflow is a natural language understanding platform that facilitates the design and integration of conversational user interfaces. It offers two editions: Dialogflow ES and Dialogflow CX. Pricing is based on the number of requests made per month. For example, Dialogflow CX charges $20 per 100 sessions for voice interactions and $5 per 100 sessions for text interactions. Assuming 10,000 text interactions per month, the monthly cost would be approximately $500, leading to an annual cost of around $6,000.

5. Custom-Built Solution on Google Cloud Platform: Our chosen approach involves developing a bespoke chatbot using Python and Flask, with data stored in an SQLite database, all hosted on GCP. The cost breakdown is as follows:

- **Compute Engine**: Hosting backend services is estimated at $10 to $25 per month, depending on usage and traffic.
- **Cloud Storage**: Storing the database and logs adds approximately $5 per month.

This results in an estimated annual cost ranging from $180 to $360.

Comparison with Ivy.ai:

Ivy.ai specializes in AI-driven chatbots tailored for higher education institutions, offering features like personalized interactions, seamless integration with existing systems, and 24/7 support across multiple channels. Pricing for Ivy.ai is customized based on the institution's specific needs, often structured around the number of full-time equivalent students. While exact figures aren't publicly disclosed, enterprise chatbot solutions typically range from $600 to $5,000

per month, depending on the complexity and scale of deployment. This translates to an annual cost between $7,200 and $60,000.

Cost Comparison:

- Custom-Built Solution: Annual cost between $180 and $360.
- Ivy.ai: Annual cost between $7,200 and $60,000.

Conclusion:

After evaluating the features and costs of each platform, we determined that a custom-built chatbot hosted on Google Cloud Platform offers the most cost-effective and flexible solution for VCU's needs. While platforms like Microsoft Bot Framework, Vercel, ChatterBot, and Dialogflow provide valuable features, their costs and integration complexities made them less suitable for this project. Ivy.ai, though offering advanced capabilities tailored for higher education, presents a significantly higher cost, making it less feasible within our budget constraints. The custom-built solution not only aligns with our financial considerations but also allows for tailored functionality specific to the VCU Computer Science department's requirements.

In developing the VCU Computer Science chatbot, we evaluated several platforms to determine the most cost-effective and functional solution. After careful consideration, we selected a custom-built approach utilizing OpenAI for natural language processing and FastAPI for backend communication due to their scalability, efficiency, and alignment with our project goals. OpenAI provides advanced AI capabilities with a pay-per-use model, while FastAPI offers asynchronous processing for handling multiple user queries efficiently. This combination ensures an optimal balance between performance, cost, and ease of integration.

OpenAi:

The primary cost for this implementation comes from OpenAI’s API usage, which operates on a pay-per-use credit system. Charges are incurred only when the API processes a request, with costs based on token usage, where both input and output tokens contribute to the total expense. The exact cost per query depends on the specific model selected, as different models have varying pricing structures. While many models remain cost-effective, typically under $0.01 per query, higher-capacity models may incur higher fees. The final model choice will be based on a balance of performance, accuracy, and budget considerations to ensure the most efficient and cost-effective solution.

FastAPI:

FastAPI is an asynchronous web framework chosen for its scalability and performance improvements over Flask. The primary cost of using FastAPI comes from hosting and infrastructure. Since FastAPI itself is open-source and free, the actual expenses are related to cloud hosting, compute resources, and potential database scaling. Hosting on Google Cloud Platform Compute Engine is estimated to cost $10–$25 per month, depending on traffic and usage. Additional costs may arise if load balancing, containerization, or API gateway services are required, but these are expected to be minimal compared to managed chatbot solutions.

Conclusion:

We have chosen to integrate OpenAI's API for natural language processing due to its advanced reasoning, contextual understanding, and ability to handle complex queries efficiently. The specific model selection will depend on factors such as cost, accuracy, and response efficiency, ensuring we balance performance with budget constraints. While OpenAI operates on a pay-per-use system, we will implement cost-control measures such as caching, optimized API calls, and prompt engineering to minimize unnecessary expenses. Additionally, the transition to FastAPI enhances the chatbot's scalability and responsiveness while keeping hosting costs manageable. This combination allows us to build a high-performance, cost-effective chatbot that meets the needs of the VCU Computer Science department.

Section J. Conclusions and Recommendations

Our chatbot project has made significant progress toward addressing the need for an accessible and reliable tool to assist undergraduate students in navigating computer science-related information. This section summarizes the progress to date and provides recommendations for the next phases of development.

Conclusions

The project began with the identification of a key problem: students often experience difficulties in accessing academic information, leading to increased workloads for advisors and delays in decision-making. To address this, the team evaluated multiple platforms and after careful consideration, a custom-built solution was chosen for its cost-effectiveness, scalability, and alignment with the project's goals.

At this stage, the our team has:

1. Transitioned from a traditional SQLite database to a Retrieval-Augmented Generation approach to enhance the chatbot's ability to pull relevant information dynamically.
2. Developed a Python-based backend, transitioning from Flask to FastAPI to support large-scale and asynchronous communication.
3. Implemented RAG to improve response accuracy by leveraging external data sources.
4. Integrated OpenAI's API for enhanced natural language processing and better handling of complex queries.
5. Built a prototype that demonstrates the ability to respond to user questions accurately.
6. Conducted initial testing to identify areas for improvement, such as response accuracy.
7. Implemented websockets to ensure multiuser functionality.

While the chatbot has made considerable progress, it is still undergoing refinement to improve accuracy, scalability, and real-world performance. The focus for the next phase of the project will be on finalizing enhancements, conducting extensive user testing, and preparing for full deployment. Additionally, we need to conduct large-scale testing to assess whether the chatbot can effectively handle multiple students using it simultaneously. All progress will continue to be documented in the project's GitHub repository to ensure a smooth transition should the project extend into the next year.

Recommendations for Future Development

As the project moves into its next phases, the following steps are recommended to ensure successful completion:

1. **Enhance Functionality**:
    - Improve the chatbot's ability to handle more complex and ambiguous queries by refining OpenAI model prompts and RAG integration.
    - Add memory features to allow the chatbot to retain context across multiple interactions, improving conversational flow.
    - Optimize query processing in FastAPI to ensure low-latency responses, even under high traffic.
2. **Expand Testing**:
    - Conduct rigorous testing with a broader range of questions to identify edge cases and improve response accuracy.
    - Release the chatbot to students for real-world testing to assess its effectiveness and ability to handle multiple users simultaneously.
    - Perform stress tests to ensure the chatbot can handle high traffic, particularly during peak times such as course registration.
3. **Refine the User Interface**:
    - Develop a more polished front-end design that is user-friendly and accessible on both desktop and mobile platforms.
    - Ensure the interface is intuitive and aligns with the VCU's branding guidelines.
4. **Gather Feedback**:
    - Encourage students and staff to interact with the chatbot prototype and provide feedback.
    - Use this feedback to make iterative improvements to both functionality and user experience.
5. **Plan for Deployment**:
    - Begin preparing for the chatbot's integration, ensuring compatibility with existing web infrastructure.
    - Work closely with VCU's IT staff to address potential challenges, such as security, authentication, and compliance with university policies.
    - Implement caching strategies to optimize API usage and reduce operational costs.
6. **Documentation and Transition**:
    - Create detailed documentation for both users and administrators, including troubleshooting steps and maintenance instructions.
    - Develop a transition plan to ensure the project can continue seamlessly into the next year, allowing the next team to build on existing work efficiently.

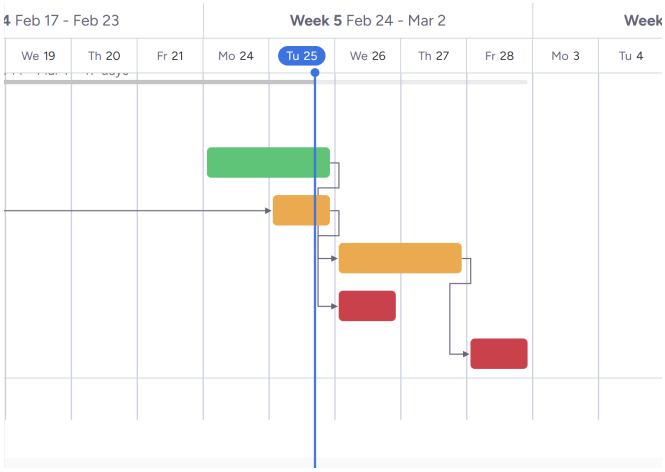
Next Steps

The focus for the next phase of the project will be on finalizing the chatbot and ensuring it is ready for full deployment. Key objectives include:

- Refining the FastAPI backend to support high concurrency and optimized query handling.
- Improving the chatbot's response accuracy through enhanced RAG tuning and OpenAI model refinement.
- Conducting real-world testing with students to evaluate performance, identify issues, and gather feedback for improvements.
- Scaling up infrastructure if necessary to support a higher volume of users.
- Documenting all progress in GitHub to facilitate future development and ensure a smooth transition if the project continues into the next academic year.

By addressing the recommendations above, the team can build a solid foundation for the final stages of development, with the goal of delivering a fully functional chatbot. The project demonstrates the team's ability to apply technical knowledge to solve real-world problems, and its success has the potential to significantly enhance the student experience at VCU. With continued progress and refinement, this chatbot can become a valuable resource for the Computer Science department and hopefully other departments in the future as well.

Appendix 1: Project Timeline



To-Do

Task	Owner	Status	Due date
Rag Implementation 1		Done	~~Jan 14~~

Subitem	Owner	Status	Date
Implement RAG using Api...		Done	Jan 14
+ Add subitem			

Testing/Updates 6		Working on it	Feb 25

Subitem	Owner	Status	Date
Combine with front end		Done	Jan 28
Test URLs		Done	
Test MultiUsers		Done	
Meeting with IT		Done	Feb 21
Add different model funci...			
Use FastAPI and websock...			
+ Add subitem			

Wrap-up 2		Stuck	Feb 26

Subitem	Owner	Status	Date
Complete Design Docum...		Working on it	Mar 25
Complete Poster			Mar 25
+ Add subitem			

Appendix 2: Team Contract

Step 1: Get to Know One Another. Gather Basic Information.

Task: This initial time together is important to form a strong team dynamic and get to know each other more as people outside of class time. Consider ways to develop positive working relationships with others, while remaining open and personal. Learn each other's strengths and discuss good/bad team experiences. This is also a good opportunity to start to better understand each other's communication and working styles.

Team Member Name	***Strengths each member bring to the group***	***Other Info***	***Contact Info***
Kennedy Martin	*Organization, communication, ML/AI course & certification, Leadership*	*Machine Learning Foundations Certificate, Java, C, Python* *Tools: pandas, numpy, matplotlib*	*martink6@vcu.edu* *discord: k_marti*
Israel Agoe-Sowah	Communication, organization,	*Java, C, Python*	agoesowahia@vcu.edu discord: izzy129
Antony Fuentes	Communication, organization, Listener	Java, C, and Python, Cybersecurity, a little bit of data science	Fuentesa2@vcu.edu Discord: antonyfuentes
Eric Simoni	Graphic design, Communication, Organization	Java, C, Python, Adobe suite	Simonie2@vcu.edu *Discord: thesuperpoop*

Other Stakeholders	***Notes***	***Contact Info***
Caroline Budwell	*Undergraduate Director of Computer Science Department*	*ccbudwell@vcu.edu*

Step 2: Team Culture. Clarify the Group's Purpose and Culture Goals.

Task: Discuss how each team member wants to be treated to encourage them to make valuable contributions to the group and how each team member would like to feel recognized for their efforts. Discuss how the team will foster an environment where each team member feels they are accountable for their actions and the way they contribute to the project. These are your Culture Goals (left column). How do the students demonstrate these culture goals? These are your Actions (middle column). Finally, how do students deviate from the team's culture goals? What are ways that other team members can notice when that culture goal is no longer being honored in team dynamics? These are your Warning Signs (right column).

Resources: More information and an example Team Culture can be found in the Biodesign Student Guide "Intentional Teamwork" page (webpage | PDF)

Culture Goals	***Actions***	***Warning Signs***
Being on time to every meeting	- *Set up meetings in shared calendar* - *Include whether meeting is in person or via discord or Zoom*	- *Student misses first meeting, warning is granted* - *Student misses meetings afterwards – issue is brought up with faculty advisor*
Informing the group of any delays in completing assignments	- *Stay up to date with each other's project responsibilities* - *Set reasonable deadlines and note when an extension is needed*	- *Student shows up for weekly meeting with no considerable work done and there was no communication with group.*
Communicate any difficulties/ask for help when needed	- *Reach out to group if there is trouble/difficulty with a task*	- *Student shows up for weekly meeting and has not completed task due to difficulty and did not reach out to group for help*

Step 3: Time Commitments, Meeting Structure, and Communication

Task: Discuss the anticipated time commitments for the group project. Consider the following questions (don't answer these questions in the box below):

- What are reasonable time commitments for everyone to invest in this project?
- What other activities and commitments do group members have in their lives?
- How will we communicate with each other?
- When will we meet as a team? Where will we meet? How Often?
- Who will run the meetings? Will there be an assigned team leader or scribe? Does that position rotate or will same person take on that role for the duration of the project?

Required: How often you will meet with your faculty advisor advisor, where you will meet, and how the meetings will be conducted. Who arranges these meetings?
See examples below.

Meeting Participants	***Frequency Dates and Times / Locations***	***Meeting Goals Responsible Party***
Students Only	*As Needed, On Discord Voice Channel*	*Update group on day-to-day challenges and accomplishments*
Students Only	*Every Thursday during class time, via discord or ERB*	*Actively work on project Documentation - Eric Simoni*
Students + Faculty advisor	*Every Tuesday 4- 4:30*	*Update faculty advisor and get answers to our questions. Status updates due Monday PM. Antony will take notes and send them out to team*

Step 4: Determine Individual Roles and Responsibilities

Task: As part of the Capstone Team experience, each member will take on a leadership role, ***in addition to*** contributing to the overall weekly action items for the project. Some common leadership roles for Capstone projects are listed below. Other roles may be assigned with approval of your faculty advisor as deemed fit for the project. For the entirety of the project, you should communicate progress to your advisor specifically with regard to your role.

- **Before meeting with your team**, take some time to ask yourself: what is my "natural" role in this group (strengths)? How can I use this experience to help me grow and develop more?
- **As a group,** discuss the various tasks needed for the project and role preferences. Then assign roles in the table on the next page. Try to create a team dynamic that is fair and equitable, while promoting the strengths of each member.

Communication Leaders

Suggested: Assign a team member to be the primary contact for the client/sponsor. This person will schedule meetings, send updates, and ensure deliverables are met.

Suggested: Assign a team member to be the primary contact for faculty advisor. This person will schedule meetings, send updates, and ensure deliverables are met.

Common Leadership Roles for Capstone

1. **Project Manager:** Manages all tasks; develops overall schedule for project; writes agendas and runs meetings; reviews and monitors individual action items; creates an environment where team members are respected, take risks and feel safe expressing their ideas.
   Required: On Edusourced, under the Team tab, make sure that this student is assigned the Project Manager role. This is required so that Capstone program staff can easily identify a single contact person, especially for items like Purchasing and Receiving project supplies.
2. **Logistics Manager:** coordinates all internal and external interactions; lead in establishing contact within and outside of organization, following up on communication of commitments, obtaining information for the team; documents meeting minutes; manages facility and resource usage.
3. **Financial Manager:** researches/benchmarks technical purchases and acquisitions; conducts pricing analysis and budget justifications on proposed purchases; carries out team purchase requests; monitors team budget.
4. **Systems Engineer:** analyzes Client initial design specification and leads establishment of product specifications; monitors, coordinates and manages integration of sub-systems in the prototype; develops and recommends system architecture and manages product interfaces.
5. **Test Engineer:** oversees experimental design, test plan, procedures and data analysis; acquires data acquisition equipment and any necessary software; establishes test protocols and schedules; oversees statistical analysis of results; leads presentation of experimental finding and resulting recommendations.
6. **Manufacturing Engineer:** coordinates all fabrication required to meet final prototype requirements; oversees that all engineering drawings meet the requirements of machine shop or vendor; reviews designs to ensure design for manufacturing; determines realistic timing for fabrication and quality; develops schedule for all manufacturing.

Team Member	Role(s)	Responsibilities
Kennedy Martin	*Project Manager*	✓ *Develops overall schedule* ✓ *Writes agendas and runs meetings* ✓ *Ensures completion of tasks* ✓ *Checks in on team members*
Israel Agoe-Sowah	Systems Engineer	✓ *Analyzes design specifications and requirements* ✓ *Coordinates and manages different project particulars* ✓ ✓
Antony Fuentes	Communications Manager	✓ *Implement communication strategies that align with the group's goals* ✓ *Primary note taker for meetings* ✓ *In charge of communicating with Advisor* ✓ *Make sure team is focused during meetings*
Eric Simoni	Logistics Manager	✓ *Contact sponsors and team members* ✓ *Update team with any new information* ✓ *Manages facility and resource usage* ✓

Step 5: Agree to the above team contract

Signature: Kennedy Martin *Date: 8/29/2024*

Signature: Eric Simoni *Date: 8/29/2024*

Signature: Antony Fuentes *Date: 8/29/2024*

Signature: Israel Agoe-Sowah *Date: 8/29/2024*

References

Provide a numbered list of all references in order of appearance using APA citation format. The reference page should begin on a new page as shown here.

[1] VCU Writing Center. (2021, September 8). *APA Citation: A guide to formatting in APA style.* Retrieved September 2, 2024. https://writing.vcu.edu/student-resources/apa-citations/

[2] Teach Engineering. *Engineering Design Process*. TeachEngineering.org. Retrieved September 2, 2024. https://www.teachengineering.org/populartopics/designprocess

[3] *The Evolution and History of AI Chatbots - Just Think AI*. (n.d.). Www.justthink.ai. https://www.justthink.ai/blog/the-evolution-and-history-of-ai-chatbots