**MS-Applied Computer Science Chatbot Project**

**Northwest Missouri State University - Applied Computer Science**

A green and white logo with a letter n

Description automatically generated

**44691** - **Graduate Directed Project I**

Under the guidance of:

Instructor: **Dr. Mark Chai**

Client: **Dr. Ajay Bandi**

**Team Bot Minds (Group2)**

* Hemanth Sai Kagitha (s559891)
* Manasa Akula (s560998)
* Trinadh Machanavazzala(s55924)
* Mahesh Kollipara (s558996)
* Teja Reddy Dasari(s547444)
* Praveen Chidipothu(s562112)

**Contents**

* Introduction
* Problem Statement
* Why Chatbot?
* Proposed Solution
* Use Cases
* Functional Requirements
* Non-Functional Requirements
* Architecture
  + Design Patterns to be Used
  + Platforms and Technologies
  + Third-Party Libraries to provide Functionality
  + Hosting Strategy and Requirements
* Data Management Plans
* Prototype Demos
  + GUI Mockups
  + Functional Prototypes
* Accomplishments
* Spring 2024 Plan
* Reference

**Introduction**

A diagram of a search engine

Description automatically generated

**Fig 1: Students Accessing the information from the MS ACS Site**

There are three categories of students (Prospective, Current, Alumni) who are trying to request information from MS-ACS website.

* 1. **Prospective students** are the students who are willing to study in MS ACS Course in Northwest Missouri State University.
  2. **Current students** are those who are already studying this course.
  3. **Alumni** who already graduated from this university.

Prospective students typically explore queries related to Admission Criteria, Course Overview, College Fee details, and Scholarship opportunities when visiting the MS-ACS site.

Current students commonly search for information regarding On-campus Jobs, Prerequisites, Electives, and other specific inquiries related to their ongoing studies.

Alumni frequently access queries about Commencement Day, Accessing Academic Transcripts, Alumni Networking, and other post-graduation concerns when visiting the MS-ACS site.

These different categories of queries reflect the diverse information needs of students at various stages of their academic journey on the MS-ACS website.

**Problem Statement**

A diagram of a search engine

Description automatically generated

A person standing next to a computer

Description automatically generated

**Fig 2: Delay in response through emails**

Prospective, current, and alumni students accessing the MS ACS website can find solutions to their queries through available resources. However, if their specific questions persist, they may need to communicate via email or inquiries to the university's designated channels. These incoming emails or inquiries will be categorized and assigned to professors, graduate assistants (GAs), or teaching assistants (TAs) based on urgency and relevance.

It's important to note that response times may vary due to the workload and different time zones of the faculty and staff members. Such delays might lead to potential frustration for students seeking prompt assistance or clarification through these communication channels.

To mitigate this, the university strives to streamline the response process and provide adequate support to address student queries effectively.

The existing communication methods on the course websites do not meet the desired standards. Several challenges have been identified:

* **Accessibility Challenges:** Some users, including students and prospects, struggle to access the website, hindering information retrieval.
* **Lack of Personalization:** The website lacks personalization, making it difficult for users to find tailored information.
* **Irregular Updates:** Information is inconsistently updated, causing uncertainty and eroding trust.
* **Complex Application Process:** The enrollment process is complex, contributing to student frustration and effort.
* **Communication Inefficiency:** Current course websites have inefficient communication methods, leading to delays in administrative responses and decreased satisfaction.

The culmination of these issues has resulted in reduced student satisfaction, creating a pressing need to re-evaluate and enhance the communication methods on the course websites.

**Why Chatbot?**

* Smart chatbot powered by AI utilizing natural language processing and machine learning.
* Emphasis on creating a user-centric experience that empowers and enriches the academic experiences of all stakeholders.
* Focus on building a supportive community to boost engagement and simplify communication.

Chat bot will be available at any time in the day so it will give the response to the queries that are asked by the students. It provides the user centric experience to different categories of students.

The chatbot makes accessing information effortless. Students can easily inquire about class schedules, tuition fees, course overview or academic policies without navigating through multiple platforms or websites. This streamlined access saves time and enhanced overall productivity. By using the chatbot, information can be accessed easily. So, the chatbot is easy to access.

**Proposed Solution**

A diagram of a chatbot

Description automatically generated

**Fig 3: Introducing the chatbot to access the information**

In response to the identified need for streamlined access to information, we have introduced a chatbot designed to facilitate easy information retrieval for students, including Prospective, Current, and Alumni. This chatbot serves as a central interface for students to inquire about various academic matters.

When a student submits a query, the chatbot processes it by searching through its database for relevant information. If the query matches available data, the chatbot promptly delivers the processed information to the student, ensuring a swift and accurate response.

However, in cases where the chatbot cannot find specific information relevant to the query in its database, it employs a fallback intent. This fallback intent is programmed to provide a response indicating that the chatbot does not currently possess the required information. This ensures transparency and prompts students to seek alternative resources or assistance if needed.

By implementing this chatbot system, our aim is to provide students with a seamless and efficient means of accessing information across various categories, ultimately enhancing their academic experience, and simplifying the information retrieval process.

**Top of Form**

**Use Cases**

A diagram of a chatbot interface

Description automatically generated

**Fig 4: Use case Diagram**

This Use Case diagram showcases the interactions between actors (the student) and the system (the chatbot) to illustrate the functionalities and processes involved.

**Use Case:** Student Query to Chatbot

**Actors:**

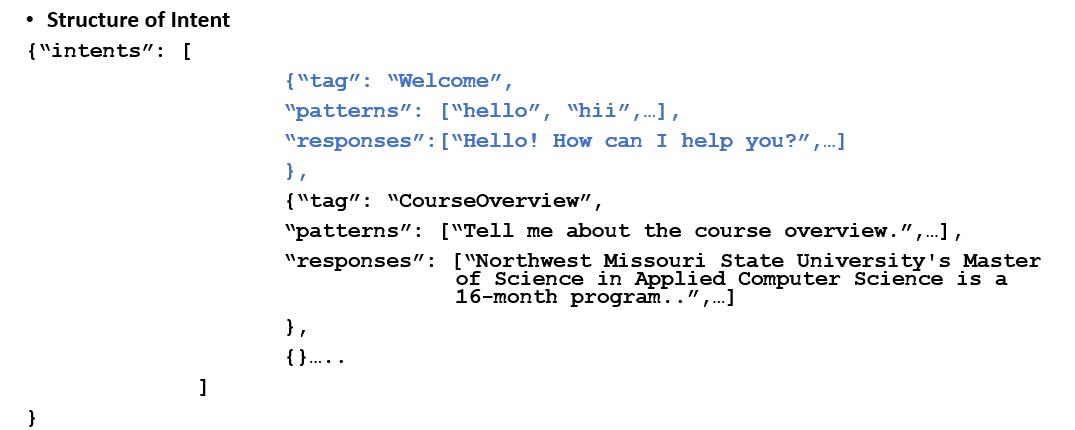
* Student: Engages with the chatbot to seek information.

**Use Cases:**

1. Request Information:
   * Student initiates a query by interacting with the chatbot.
   * The chatbot receives and processes the query.
2. Access Data:
   * The chatbot accesses the data source (e.g., data.json file) containing relevant information.
3. Retrieve Information:
   * The chatbot searches the data source for the necessary information based on the student's query.
4. Provide Response:
   * If relevant information is found:
     + The chatbot formulates a response based on the retrieved data.
     + It delivers the response to the student, providing the necessary information.
5. Fallback Response:
   * If the chatbot cannot find relevant information in the data source:
     + It triggers a fallback response mechanism.
     + The chatbot informs the student that it doesn't currently possess the requested information.

In the context of the described use case, the data.json file serves as the primary data source for the chatbot, organized in the form of intents. An intent represents a specific task, action, or purpose that a user seeks to accomplish or inquire about when interacting with the chatbot.

The data.json file is structured to contain these intents, providing the chatbot with categorized information that aligns with the various queries or tasks students might request. Each intent within the JSON file comprises key-value pairs or structured data representing the following components.



**Fig 5: Structure of Intent**

**Intent Structure:** An intent consists of three main parts:

1. **Tag:** The tag serves as the root element used by the chatbot to determine the relevance of a query. It acts as a label or identifier for the particular category of information or action associated with the intent.
2. **Patterns:** This section contains a set of questions, phrases, or patterns that users might input when seeking information or performing a task related to the intent. These patterns help the chatbot recognize user queries that align with the intent.
3. **Responses:** Within an intent, the responses section includes a set of predefined answers or replies that the chatbot can provide when triggered by a user query associated with that intent. These responses are designed to address the user's query or provide relevant information based on the intent's tag.

This structure allows the chatbot to categorize queries based on tags, recognize patterns or questions from users, and respond with appropriate answers or information associated with the respective intent. Intents can be expanded or added as needed to accommodate different user queries and scenarios.

**Functional Requirements**

* The chatbot must store chat history and feedback history.
* The chatbot should answer the questions related to three categories.
* The chatbot must process and handle user queries, requests, and commands effectively.
* The chatbot must recognize user intents to determine the user's purpose or request.
* The chatbot must generate contextually relevant and accurate responses to user queries.
* The chatbot must retrieve information from databases, APIs, or external sources as needed to provide answers and services.
* The chatbot shall handle errors, exceptions, and unexpected user inputs, providing clear error messages and suggestions.

By considering the above functional requirements our proposed chatbot will be able to provide the information to the users in a proper way which helps to get rid of problems faced by the students.

**Non-Functional Requirements**

1. **User-Friendly and Intuitive Interface:**

A user-friendly chatbot interface ensures that users can navigate and interact with the system without complications. This involves a clean design, clear instructions, and easily accessible functionalities. For instance, having intuitive menus, prompts, and guided interactions can help users feel more comfortable and confident while using the chatbot.

1. **User Feedback Mechanism:**

Providing a dedicated feature within the chatbot interface for users to report issues or share their opinions helps in gathering valuable insights. This could be a feedback form, a suggestion box, or a direct link to customer support. Allowing users to express their concerns or suggestions not only encourages engagement but also assists in understanding user expectations and areas that need improvement.

1. **Continuous Training Material Improvement:**

Ensuring the chatbot's training materials are regularly updated and enhanced based on user feedback and evolving needs is crucial. This involves monitoring user interactions, analyzing feedback, and using this data to refine the chatbot's knowledge base. By adapting to user queries, introducing new topics, and refining responses, the chatbot remains relevant and efficient in addressing user inquiries effectively over time.

**Architecture**

A diagram of a chatbot

Description automatically generated

**Fig 6 : Architecture Diagram**

The architecture diagram illustrates a seamless interaction between students and the chatbot. Upon receiving a query from a student, the chatbot employs Lemmatization, breaking down the query into base forms for better understanding. These modified tokens are then compared with third-party libraries to align the query with available information. If discrepancies arise, the chatbot uses library data to refine the tokens, ensuring accuracy. Using this refined data, the chatbot formulates a tailored response, providing the student with precise and relevant information. This process helps the chatbot understand queries effectively and offer accurate responses, aiding students in resolving their issues efficiently.

**Design patterns to be used**

**Lemmatization:**

A diagram of a computer system

Description automatically generated

**Fig 7: Lemmatization Process**

Steps involved in the Lemmatization Process:

1. **Input Preprocessing**: The student's text query undergoes cleaning, tokenization (breaking into words), and Part-of-Speech tagging for grammatical identification.
2. **Lemmatization Engine and Library Use:** The Lemmatization Engine efficiently employs libraries like WordNet/punkt. It identifies the base forms (lemmas) of words in the query, ensuring standardization of different word forms.
3. **Base Form Output:** The output provides the base form of each word in the query, aiding in standardization (e.g., "running" to "run").
4. **Utilization in Natural Language Processing:** Lemmas are crucial for further Natural Language Processing tasks like Intent Recognition. They help in understanding the query's meaning, enabling accurate interpretation of the user's intent.

Below is the example of lemmatization process:

A white sheet with black text

Description automatically generated

**Fig 8: Example of Lemmatization**

**Platforms and Technologies**

Below mentioned are the tool and technologies used in our project.

**Python:** Python is a versatile and high-level programming language known for its simplicity, readability, and extensive community support. Python's syntax is designed to be clear and readable, emphasizing code readability and reducing the cost of program maintenance.

**Natural Language Toolkit (NLTK):** Natural Language Toolkit (NLTK) is a powerful library in Python for working with human language data. It provides easy-to-use interfaces to work with linguistic data and perform various natural language processing (NLP) tasks.

**SQLite:** SQLite is a lightweight, serverless, self-contained, and open-source relational database management system (RDBMS). SQLite is serverless, meaning it doesn't require a separate server process to operate. The entire database engine is implemented as a single, self-contained library that can be embedded directly into applications. SQLite requires minimal setup and configuration. **Top of Form**

A group of logos on a white background

Description automatically generated

**Fig 9: Tools and Technologies**

**GitHub:** GitHub is a web-based platform and service that provides hosting for software development and version control using Git. It offers a collaborative environment for developers to work on projects, track changes, and manage code repositories. GitHub is widely used by developers, teams, and organizations for version control and collaboration.

**Jupyter Notebook:** Jupyter Notebook is an open-source, web-based interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports multiple programming languages.

**VS Code:** Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It has gained widespread popularity among developers due to its lightweight design, extensibility, and support for a wide range of programming languages.

**WordNet:** WordNet is a lexical database of the English language that groups words into sets of synonyms called synsets, providing short, general definitions and listing semantic relationships between these synonym sets.

**Third-Party Libraries to provide Functionality**

The required functionality is provided using third-party libraries such as **WordNet** and **Punkt**.

**Wordnet:** It serves as a lexical database, capturing relationships between words based on their meanings rather than just their alphabetical order.  
It Contains 155,327 words organized in 175,979 synsets.  
A synset or synonym set is defined as a set of one or more synonyms.

**Example**: WordNet synsets for the word “hello”

**Synset :** hello, hullo, hi, howdy, how-do-you-do  
**Definition:** An expression of greeting.  
**Example Usage:** "She greeted him with a cheerful hello."

**Punkt:** The Punkt module is a component of the Natural Language Toolkit (NLTK), a powerful library for natural language processing (NLP) in Python. Specifically, Punkt refers to the Punkt tokenizer, a pre-trained unsupervised machine learning model included in NLTK. punkt is designed to learn parameters in Natural Language Processing. It identifies the boundaries between sentences (multiple) in text. It enables to analyze the sentence for tasks like sentiment analysis, intent recognition, or information extraction.

**Example:**

text = “This is a sample sentence. And here is another one!”

[‘This is a sample sentence.’, ‘And here is another one!’]

It returns list of sentences from input query

**Hosting Strategy and Requirements**



**Fig 10: Heroku**

Our chatbot will be hosted on Heroku, providing a stable and reliable online service to enhance the user experience. Heroku is a cloud platform that provides a service for deploying, managing, and scaling applications. It is designed to make it easy for developers to build, deploy, and scale applications without worrying about the underlying infrastructure. Heroku supports multiple programming languages and frameworks, allowing developers to focus on writing code and delivering features rather than managing servers.

Here are key features and aspects of Heroku: Platform as a Service (PaaS), Multi-Language Support, Containerization, Developer Productivity, Scalability, Add-ons and Services, Database Support, Continuous Integration and Deployment (CI/CD),Logging and Monitoring, Collaboration and Teams, Free Tier.

**Data Management Plans**

**Data Storage and Organization:**

* Using JSON files for initial intents and response storage.
* Plan to transition to a dynamic database (SQL/NoSQL) for scalable data management.

**Data Processing and Optimization:**

Regular data cleaning and preprocessing to maintain quality. Data processing involves collecting, organizing, and manipulating raw data to produce meaningful information for decision-making and analysis. Optimization refers to the improvement of processes, systems, or algorithms to enhance efficiency, reduce resource usage, and achieve better performance.

**Backup and Recovery:**

Routine data backups to prevent loss. Backups act as a safety net, protecting against accidental deletions, hardware failures, software bugs. In the face of disasters, whether natural or technological, having a backup and recovery plan ensures that critical data can be restored.

**Prototype Demos**

**1.GUI Mockups:**

TKinter is primarily used for creating GUI (Graphical User Interface) applications in Python, and these applications are generally designed to run on a local machine.

**A screenshot of a computer chat

Description automatically generated**

**Fig 11: GUI Mockup**

1. **Functional Prototypes -1**

The source code can be saved as python script (.py), and can be run in console or terminal, But the jupyter notebook facilitates this to run the code for the Command Line Interface.

A screenshot of a computer

Description automatically generated

**Fig 12: Functional Prototypes-1**

**Functional Prototypes -2**

This is a web interface developed using Flask library, that serves as a background framework that serves user-centric experience.

**A screenshot of a computer

Description automatically generated Fig 13: Functional Prototypes-2**

**Accomplishments**

**Functional Chatbot:** Successfully developed a functional chatbot that can understand user queries and provide relevant responses.

**User Interface Design:** Created a user-friendly interface for the chatbot, making it easy for users to interact with and receive information.

**Natural Language Processing (NLP):** The chatbot demonstrates NLP capabilities by understanding and responding to natural language inputs from users.

**GUI and Web Interface Implementation using Flask:** Successfully implemented a graphical user interface (GUI) and a web interface for the chatbot using Flask.

**Spring 2024 Plan**

In GDP-2, we will be integrating Neural Network Model with Generative AI.

* With this, The Chatbot can be coherent, contextually relevant, and varied responses.
* The combination of neural networks and generative AI models empower the chatbot to engage in small talk, making interactions more dynamic.
* Add more intents using suitable frameworks/APIs.
* Integrate the chatbot within a website, moving beyond a basic web interface.
* Develop a sample website for testing the chatbot.

**References**

1. [**https://www.researchgate.net/publication/327667155\_ChatBot\_For\_CollegeManagement\_System\_Using\_AI**](https://www.researchgate.net/publication/327667155_ChatBot_For_CollegeManagement_System_Using_AI)
2. [**https://www.scribd.com/document/511798296/Chat-Bot-for-College-Management-System-U**](https://www.scribd.com/document/511798296/Chat-Bot-for-College-Management-System-U)
3. [**https://www.ijnrd.org/papers/IJNRDA001003.pdf**](https://www.ijnrd.org/papers/IJNRDA001003.pdf)
4. [**https://ieomsociety.org/proceedings/2022istanbul/704.pdf**](https://ieomsociety.org/proceedings/2022istanbul/704.pdf)
5. [**https://www.mdpi.com/1999-5903/15/8/260**](https://www.mdpi.com/1999-5903/15/8/260)

**GitHub Repository URL:** [**https://github.com/TrinadhM-dev/GDPProject-02**](https://github.com/TrinadhM-dev/GDPProject-02)

**GitHub Final\_Submission in repo:**

[**https://github.com/TrinadhM-dev/GDPProject-02/tree/main/final\_submission**](https://github.com/TrinadhM-dev/GDPProject-02/tree/main/final_submission)