Technological Feasibility

JabberJack

Team Member:

Sara Harris, Tyler Zimmerman, Jiasheng Yang, Gabriel Proctor

Team Mentor: Felicity H. Escarzaga

Client: Dr. Andy Wang

Northern Arizona University

---

3 November, 2021

Contents

1.0 Introduction

JabberJack

Chatbots and other helpful bots have become increasingly common throughout both academia and in the business world [1]. Many websites implement helpful bots that can answer basic questions about products and other services. Even here at Northern Arizona University (NAU) there is a chatbot already in place on the Campus Health Services site [2]. Chatbots are not as advanced as people may believe them to be, but there is a great deal of research teams working to improve chatbots in general [1]. Data is the driving force behind chatbots, and without it, they could not operate on the level that they do currently. Chatbots become more helpful the more data that they have at their disposal [4]. Usually, this data is contained within a database or in some central location that the chatbot can grab from. Data and information do not just help chatbots; they also allow the world of academia to function.

Here at Northern Arizona University there are thousands of students and faculty who access NAU systems daily [3]. There is already a chatbot in use at NAU [2] but it is specialized to only know how to answer questions about Campus Health Services. The project's client Dr. Andy Wang wants to create a chatbot that will answer student and visitor questions in a timely manner. Dr. Andy Wang is the dean of the College of Engineering, Information systems, and applied sciences. Dr. Andy Wang is in the business of people and wants to improve the quality of life for the people here at NAU. A chatbot would provide answers to questions in a timely manner to students and visitors as well as alleviate stress from the people who had to previously answer those questions.

While the chatbot has already been used to deal with some students' services at NAU, it is still difficult to answer users' questions easily and intelligently. Users face some problematic business functions during conversing with the chatbot:

- The difficulty of getting answers
    - These questions are not easily answered or found in a centralized location online.
    - Data required to answer these questions is often buried under multiple levels of websites and search queries.
- The lack of database
    - There is no centralized database where a chatbot can grab Questions and Answers (QA) from; the data is too scattered.

- The lack of user information
    - The staff at Northern Arizona University work hard to perform a long list of tasks but it is difficult for them to address students and visitor questions.

ChatterJack would provide that centralized database. ChatterJack will be able to answer questions in a timely manner and be able to answer a variety of questions from students and visitors. While the questions will currently be limited to the college of engineering, there are plans to add the structure required to add other information to the ChatterJacks' base knowledge. This project will provide a platform that will consolidate Northern Arizona University's data into a single spot so that students and visitors do not have to chase down answers on the internet or ask Northern Arizona University staff. Rather than digging through NAU's stack of websites for minutes on end, a simple question to the ChatterJack chatbot will provide a user with the information they need in seconds. ChatterJack will have these core features:

- A comprehensive and centralized database.
- Reasonable and intelligent response times to the users' questions.
- Upload the user information from NAU's data securely.

This technological feasibility document will explore what challenges the project will face as well as solutions that the project will utilize in order to produce a chatbot that will efficiently and effectively answer the questions that users may have.

2.0 Technological Challenges

JabberJack

- **Authentication and Security**
    - The Chatterjack chatbot will need to protect user information and will need to be able to authenticate those users. Authentication will allow users to log in to the chatbot and access data associated with it. Security will protect the users' information and the chatbot's data from unauthorized access.
- **Database Management System**
    - The key task of the chatbot is providing answers to users' questions. There should be a reliable and extensible database management system so the Chatterjack can store and update questions and answers.
- **Searching Algorithms**
    - The Chatterjack will require an efficient algorithm that can filter through large quantities of data. The algorithm needs to be able to traverse a database quickly and return correct answers to user questions.
- **User Interface (UI)**
    - The user interface should provide users with a professional looking way to interact with the chatbot. The chatbot needs a way to create this UI and it needs to allow users access to both the chatbot and its data.

A chatbot requires a plethora of information and functionality to be coherent and usable in the real world. Administrators that will maintain the chatbot will need a way to login and access the data. For this, there needs to be security and authentication to keep out unauthorized access. The chatbot will need to be able to answer pre-inputted questions that users have. In order to accomplish this the chatbot will need a database that will contain pre-determined questions as well as corresponding answers while maintaining the ability for new information to be added. an algorithm will be utilized to scan the database every time a user asks the chatbot a question. Finally, a user interface will need to act as a delivery system. The user interface will provide a space where users can interact with the chatbot. This interface should be simple and visually appealing to provide a user friendly experience. The eventual goal is to integrate the technology into a physical robot.

This document will explore the options available to solve the given challenges of: authentication and security, database management systems, algorithms, and user interfaces. In each analysis section the options will be rated on a scale from 1 - 5. Where a 5 means that the

option extremely satisfies the criteria, a 4 means that the option highly satisfies the criteria, a 3 means that the option moderately satisfies the criteria, a 2 means that the option slightly satisfies the criteria, and a 1 means that the option does not satisfy the criteria at all.

3.0 Technological Analysis

JabberJack

The following sections analyze and compare various technologies and methods that may be used to solve the technological challenges. Each analysis contains the introduction, desired characteristics, alternatives, analysis, the chosen approach, and the feasibility of the chosen approach.

3.1 Authentication and Security

User authentication and security are central to any website that uses login authentication. For the purposes of this project, user authentication is necessary to allow authorized users access to the chatbot's data. Thus allowing authorized users the ability to make changes and updates to the chatbot's base knowledge. The only users that should be able to access this data should be administrators. This will secure the chatbot and its coupled data.

3.1.1 Desired Characteristics

The following characteristics will be the guide to what approach is chosen:

- **Security**: It is important to keep user information secure so that unauthorized users cannot access the data that drives the chatbot. The user information should be secure and inaccessible to outsiders.
- **Access**: Users/Administrators should be able to login and obtain access to the data that drives our chatbot. There needs to be a way to create new administrators for the bot. This login should allow administrators complete access to the chatbot and allow them to manipulate the data within it; specifically they should be able to add new questions, answers, and change other data pertaining to the bot itself.
- **Ease of Implementation**: This module of the software should use something that is easily understandable and should have intensive documentation on how to utilize tools provided by the module.

3.1.2 Alternatives

There are several ways that a login page can be created with secure information as well as enable access to the software. Here are some of the options that this section will explore:

- **Web Application Firewall (WAF)**: A web application firewall or WAF is a shield that is put in front of a web application that protects it from malicious attacks. Many WAFs protect against attacks such as: cross site forgery, cross site scripting, file inclusion, and SQL injection attacks. WAFs operate through a set of policies that protect against vulnerabilities by filtering and monitoring HTTP traffic [7]. There are some open source WAFs as well as proprietary WAFs that require either subscription or licensing agreements, the latter is more common [7].
- **PHP**: PHP stands for "Hypertext preprocessor" and it is a scripting language that is used on websites like Facebook and Yahoo. A scripting language is mostly used for routine tasks and is usually executed without compilation [10]. PHP can provide and improve website functionality. Some examples of these added functionalities are: user access management and the encryption of data [8]. PHP pairs really well with database management systems like postgreSQL and MySQL.
- **JSON Web Tokens (JWT)**: A JSON web token or JWT is a json-encoded representation of a claim that can be transferred between two parties [9]. It is a mechanism that can verify the owner of some JSON data and is a URL safe string that can contain an unlimited amount of data that is "signed" by the owner. A JWT can be trusted because once it is sent it cannot be modified by a third party which makes it a secure way to move data; although it is vulnerable to other attacks like a cross site scripting attack [9].

3.1.3 Analysis

In this section the options will be evaluated based on the desired characteristics listed in section 3.1.1. This section will follow the ranking system laid out in the last paragraph of section 2.0.

- **Web Application Firewall (WAF)**
    - **Security**: A web application firewall or WAF provides great security for not just the data but for the entire web application. For this reason it receives a **5 out of 5** for security.
    - **Access**: Web application firewalls can provide great user authentication and session tracking. Most WAFs, but not all, can block unauthorized access and track access attempts. For these reasons it receives a **4 out of 5** for access.

- **Ease of Implementation**: Web application firewalls are complicated as they can do so much and it would be difficult for some new to the technology to implement it correctly. For these reasons it receives a **1 out of 5** for the ease of implementation.

The overall score for WAF **3.3 out of 5**, this is determined by the average of the previous scores.

- **PHP**
  - **Security**: PHP can provide great security against attacks like SQL injection attacks, cross site scripting attacks, and even file inclusion attacks. Since PHP is able to protect against a wide range of attacks it receives a **5 out of 5** for security.
  - **Access**: PHP is able to create a fully functional login page that can track by session and can accurately authenticate users [6]. For that reason PHP receives a **5 out of 5** for access.
  - **Ease of Implementation**: PHP is a very well documented scripting language and there is a large community of users. But PHP is a difficult language to learn according to many Quora users who are advanced in PHP [5]. Since PHP is so common and has extensive documentation it receives a **4 out of 5** for ease of implementation.

The overall score for PHP **4.7 out of 5**, this is determined by the average of the previous scores.

- **JSON Web Tokens (JWT)**
  - **Security**: JSON web tokens are secure and once sent cannot be altered by any third parties. But JWTs can be intercepted and read by third parties and can expose user data [9]. There are lots of features and implementing them correctly is a difficult task that often leads to mistakes [9]. For these reasons JWTs receive a **3 out of 5** for security.
  - **Access**: JWTs can be used to authenticate users and can give users access to restricted sections of websites [9]. This is very effective and these tokens can be set to expire thus invalidating a user [9]. For these reasons JWTs receive a **5 out of 5** for access.
  - **Ease of Implementation**: Since the scope of JSON web tokens are so large and they can be used to do so many different things it is difficult to correctly implement them [9]. Since JSON web tokens are difficult to use correctly JWTs receive a **3 out of 5** for ease of implementation.

The overall score for JWT **3.7 out of 5**, this is determined by the average of the previous scores.

3.1.4 Chosen Approach

There are several options that the project can implement to accomplish authentication and security; Web application firewalls (WAFs), PHP, or JSON web tokens (JWTs). Using the desired characteristics mentioned in section 3.1.2, table 3.1.1 provides each option in section 3.1.3 with a ranking between 1-5.

Table 3.1.1: Authentication & Security Rankings

Options	Security	Access	Ease	Average
Web Application Firewall(WAF)	5	4	1	3.3
PHP	5	5	4	4.7
JSON Web Tokens(JWT)	3	5	3	3.7

According to table 3.1.1 the best choice is PHP. It has great security with minimal configuration and can protect against the majority of attacks. PHP also can provide login pages that can authenticate users in a timely manner [10]. While PHP is not an easy language to learn, it is extremely common and there is a very large community and documentation to support the implementation (Quora 2018). The other options, Web application firewalls (WAFs) and JSON web token (JWTs), both have their shortcomings and do not completely satisfy all of the desired characteristics laid out in section 3.1.1. Web application firewalls are not simple and do not completely satisfy the access requirements for the project, not all WAFs support user authentication [7]. JSON web tokens are a great resource for providing user authentication but JWTs fall short in ease and security; they are not easy to implement due to their large scope and are not very secure as they can be accessed and viewed by third parties [9].

3.1.5 Proving Feasibility

The option of PHP will provide the project with appropriate authentication and security to protect user data and the application. Moving forward, implementing a basic login page with at least one user can be testable and can show that PHP provides adequate authentication for the application. PHP can protect against SQL injection attacks, and this can be shown utilizing a small script that scrubs data and does not allow unauthorized access.

3.2 Database Management System

Databases are the backbone of how every information based application stores and retrieves data. The limitations of the language used to implement a database will determine how efficient the chatbot can be at retrieving pertinent information to the users' queries. With that, there are several different ways to structure a database, each with its own benefits and shortcomings. Given the context of the project the difficulty lies in what types of structures should be used to leverage those benefits and negate as many of those shortcomings as possible.

3.2.1 Desired Characteristics

A chatbot needs to be able to handle complex interactions associated with human speech and sentence structure and in a timely manner. The following are the desired characteristics of what a chatbot database management system should have:

- **Ease of Implementation**: It is important to take into account how difficult it is to utilize database management software. Database management systems all have features that help developers meet the needs of their projects. But as these tools become more complex, it also becomes more difficult to use. Having more features is good, but not so much that it becomes a hindrance.
- **Speed of Query Resolution**: This is the measure of how quickly the database management system handles SQLqueries, where higher speed is desirable. This metric is not based on any specific numbers but rather an alternative's speed in relation to other alternatives. This speed determines how quickly and efficiently the chatbot can retrieve an answer to a question without taking into account any search algorithms needed to determine which database registry needs to be accessed.
- **Scalability**: This is the measure of how well language handles databases that get larger. When a database is small, for example, less than 1000 entries, this measure is negligible. But as more and more questions are asked and added to the database the number of entries could grow larger. Knowing how well a given management system handles larger datasets will help determine how efficient data retrieval will be as more data is attained.

3.2.2 Alternatives

- **MySQL**: This is traditionally a server based database management system, but can be adapted in a local setting. MySQL is a relational database management system meaning a typical schema (the schematic of the actual structure of the database) has multiple tables linked together with related information.
- **SQlite**: This alternative is a serverless management system. It has the database management functionality fully integrated into the end product [REF]. SQLite is also a relational database management system so follows a similar structure to mySQL.
- **PostgreSQL**: This alternative is a server based management system but like other alternatives can be adapted to a local environment. The key difference from this alternative to others is that postgreSQL is an object oriented relational database management system. This means it supports objects, classes, inheritance and other functionality typically found in object oriented programming languages.

3.2.3 Analysis

In this section the options will be evaluated based on the desired characteristics listed in section 3.2.1. This section will follow the ranking system laid out in the last paragraph of section 2.0.

- **MySQL**
    - **Ease of Implementation**: Due to the popularity of MySQL and the abundance of tutorials online, this database software is relatively easy to use. The website associated with this alternative claims "3 minutes from download to development"[12]. This claim assumes some familiarity with SQL as well as it being in its server based configuration. Converting to a local database will be slightly harder. For these reasons mySQL scores a **4 out of 5**.
    - **Speed of Query Resolution**: While sql query resolution speed is also affected by the complexity of the query itself there can be generalities that can be generated. MySQL in comparison to another alternative, MySQL is faster at read only transactions, where it is unnecessary to change the information within an entry. MySQL does not do any native query optimization, but can be implemented by a developer. Considering that a chat bot typically does not have to change an answer after it has been found that read only transactions are more important than read-write transactions. MySQL's query resolution does not change its speed significantly with larger datasets. For these reasons mySQL scores **5 out of 5**.
    - **Scalability**: MySQL can be scaled easily with relatively minor speed impacts. With the official documentation putting mySQL's upper limit of database size being limited by the operating system it is running on rather than any internal

limits. With most modern computers allowing for terabytes worth of information, it is safe to conclude that MySQL scales well, and as such scores **5 out of 5**.

The overall score for MySQL **4.7 out of 5**, this is determined by the average of the previous scores.

- **SQLite**
    - **Ease of Implementation**: Due to its serverless nature, SQLite is even easier to implement than MySQL, but with the various drawbacks in comparison such as its lack of data integrity (SQLite has no native ability to fully prevent improper data from being inserted into tables) this extra work will be on the part of the programmer, and as such receives a **3 out of 5**.
    - **Speed of Query Resolution**: SQLite is faster at retrieving data in comparison to most server based architectures. This is because SQLite is a serverless implementation and as such does not have any of the overhead associated with network communications. With this in mind SQLite receives a rating **5 out of 5.**
    - **Scalability**: According to TablePlus.com, SQlite requires a tremendous amount of memory as the size of the database increases, and similarly to MySQL, does not have any native concurrency further reducing ability to scale. This alternative scores a **3 out 5**.

  The overall score for SQLite **3.7 out of 5**, this is determined by the average of the previous scores.

- **PostgreSQL**
    - **Ease of implementation**: The implementation of PostgreSQL is considered comparable to that of MySQL although with less online guides. Postgre SQL also implements a sharper definition for SQL queries, and as such requires more precision on the part of the developer. Given postgreSQL's object oriented nature, the extra features could make implementation more complex and possibly more difficult. With these factors in mind this alternative scores **3 out of 5.**
    - **Speed of Query Resolution**: In terms of individual transactions PostgreSQL falls behind that of MySQL and SQLite in read only transactions, but is faster in read-write transactions. Considering read-write transactions will not occur as frequently as read-only transactions, the benefit of read-write transactions is outweighed by its read-only shortcoming.Thus receiving a score of **4 out of 5** .
    - **Scalability**: PostgreSQL scales much easier than SQLite. According to postgreSQL's official documentation the upper limit of database size is "unlimited"[13] and as such can be assumed to rely on the hardware itself to determine the limits. For these reasons PostgreSQL receives a **5 out of 5**.

  The overall score for PostgreSQL **4.0 out of 5**, this is determined by the average of the previous scores.

3.2.4 Chosen Approach

Using the desired characteristics mentioned in section 3.2.2, table 3.2.1 provides each option in section 3.2.3 with a ranking between 1-5.

Table 3.2.1: Database Management System Rankings

Options:	Ease of Implementation	Speed of Query Resolution	Scalability	Average
MySQL	4	5	5	4.7
SQLite	3	5	3	3.7
PostgreSQL	3	4	5	4.0

As shown in table 3.2.1, MySQL had the highest average amongst the candidates. In comparison to the other it is clear that mySQL is the most appropriate database management system. In terms of its speed, SQLite is comparable, and in terms of scalability postgreSQL is comparable. Only mySQL puts both of those benefits into one package while also remaining relatively easy to implement.

3.2.5 Proving Feasibility

In order to prove mySQL is the appropriate choice for the chatterjack, a simple demo database of user questions and answers can be created, as well as a simple framework for usernames and passwords in a separate database managed by the same system. These demos should prove that mySQL is able to accommodate the requirements for all other sections of the project.

3.3 Searching Algorithms

A search algorithm is used to traverse a set of data and find specific patterns within the data. In this project the data that the chatbot is receiving are user questions. This search algorithm should be able to recognize words and match them to the pre entered questions contained in the chatbot's database. Then the algorithm should grab the answer associated with the question asked. The algorithm should be fast enough to provide an almost instantaneous answer to users.

3.3.1 Desired Characteristics

The following characteristics will be the guide to what approach is chosen:

- **Searching Speed**: When utilizing a chatbot users expect instantaneous answers to the questions that they ask. A high searching speed is important in order to provide that instant response. When the question asked by users requires the chatbot to traverse a large quantity of data, the search time needs to be low enough that users are not kept waiting.
- **Space**: If a search algorithm requires a large amount of space it will impair the search speed. An algorithm that utilizes a small amount of space is optimal so that it can function at the highest level possible.
- **Accuracy**: The completion of the task depends on the accuracy of searching corresponding questions from the saved dataset. For example, if the searching result is a similar question rather than the same, the answer may not be what the customer wants.

3.3.2 Alternatives

An efficient string-searching algorithm will help the Chatterjack traverse the database quickly. The Chatterjack will utilize a database of questions and answers by matching user questions to corresponding questions in the database.

- **Knuth Morris Pratt Pattern Search (KMP)**: The Knuth Morris Pratt pattern search algorithm is a common pattern matching algorithm. It works by comparing two arrays of data and then outputting the matching pattern [15]. It is basic in the sense that it does not utilize any complex coding techniques. It simply matches patterns up in the two arrays. For the purpose of this project one array would be the user question and the other would be the questions that are contained within the database itself.
- **Deterministic Finite Automata algorithm (DFA)**: The DFA algorithm functions based on a deterministic finite automaton where the algorithm will either accept or reject a given string. It works by organizing the data set into a tree structure to better locate

specific keys or in this case questions and then return corresponding answers. This tree is called a trie tree. This tree supports find and insert operations.

- **Aho-Corasick algorithm (AC)**: The Aho-Corasick algorithm or (AC) is another pattern finding algorithm that can search for multiple different patterns within a string. It utilizes a string search similar to that of the Knuth Morris Pratt Pattern search algorithm and utilizes a trie tree much like that of the deterministic automaton based algorithm.

3.3.3 Analysis

In this section the options will be evaluated based on the desired characteristics listed in section 3.3.1. This section will follow the ranking system laid out in the last paragraph of section 2.0.

- **KMP**

  KMP utilizes two arrays to record the position pointer of string (i) and pattern (j), respectively. The core concept of the Knuth Morris Pratt pattern search algorithm is that it utilizes two separate pointers and at every slot in the array compare to see if it matches the data in the other array's slot [15].
  - **Searching Speed**: Only recording the useful information, while avoiding the useless backtracking that always starts from the last position after a mismatched position, gives an edge on improving the searching speed. However, the run time depends on where the pattern is located and how long and what is the length of the input question. For these reasons, this alternative scores **3 out of 5** in this category.
  - **Space**: The algorithm needs to set up two spaces for storing two array pointers, the primary string and pointer. And if there is a long input question or the pattern appearing in the back of the sentence, it might consume large amounts of memory. As a result of this potential complication, this alternative scores **3 out of 5** in this category.
  - **Accuracy**: The accuracy is dependent on the length of input question or the position of the pattern within the input string. This algorithm might become stuck in a loop. As a result of this potentially catastrophic problem, this alternative scores **3 out of 5** in this category.

  The overall score for KMP **3 out of 5**, this is determined by the average of the previous scores.

- **Variants of KMP - DFA & AC**

This section will be an analysis of DFA and AC by utilizing two demo scripts, "DFA_demo.py" and "AC_demo.py", By testing the total search time associated with the same question, "Could

you tell me more information about CEIAS?", and retrieving the "demo_keywords.txt" that contains five example patterns, "brain", "JabberJack", "LumberJack", "NAU", "CEIAS".

The result of the demo will record how many patterns are matched and replace the matched pattern by "*". Here are the results of demos:

Figure 3.3.3.1 DFA_demo

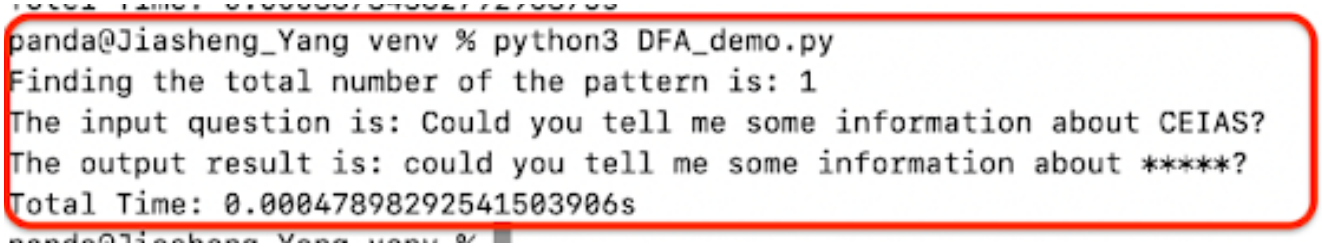

Figure 3.3.3.2 AC_demo

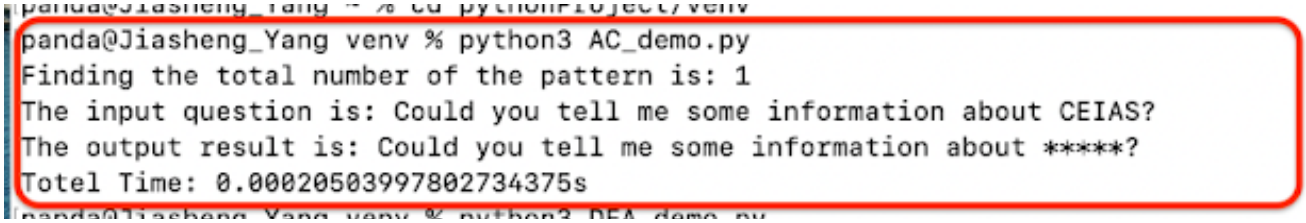

Table 3.3.3 Runtime

String-searching Algorithm	Runtime
DFA	0.000479
AC	0.000205

According to Table 3.3.3, Runtime, the searching speed of AC algorithm is faster than DFA algorithm.

- **DFA**
  The core concept of DFA algorithm is setting up a data structure called Trie Tree to store the questions by lexicographical order [19]. It utilizes the current state and current string as a triggering event to judge whether the user question is in the database or not. For example, if the pattern and primary string are "ABC", and DFA starts from NULL, when it receives event "A", it will go to "A". After that, when it receives event "B", it will go to "AB", until it matches the whole pattern.

- **Searching Speed**: Searching the pattern by prefix from Trie Tree is good at avoiding useless backtracking. However, if the prefix of the pattern is very similar, it still needs much backtracking. In addition, it takes a little time to set up the Trie Tree. For these reasons, this alternative scores **3 out of 5** in this category.
- **Space**: It needs to set up the space to store Trie Tree, which takes a considerable amount of memory. With that said however, there is no need for two position pointers. As a result, this alternative scores **3 out of 5** in this category.
- **Accuracy**: Searching the Trie Tree which organizes by lexicographical order is more like retrieving a word in the dictionary; if the question is in the saved dataset, it can always return the relative answer. To sum up, this alternative scores **5 out of 5** in this category.

The overall score for DFA **4 out of 5**, this is determined by the average of the previous scores.

- **AC**

The core concept of AC algorithm is that it combines the KMP and DFA algorithms. Which takes advantage of the double pointer/array structure of the KMP algorithm and Trie Tree data structure that is important to the DFA algorithm[16].

- **Searching Speed**: Searching the pattern by prefix from the database of questions and recording the mismatched pointers makes the AC algorithm the optimal string-searching algorithm. It is capable of searching the database of QA in the shortest time whether it goes through the first match or multiple matches. To sum up, this alternative scores **5 out of 5** in this category.
- **Space**: Generating the Trie Tree occupies the huge space since there is the huge dataset of QA so that JabberJack could answer as many questions as possible. In addition, it needs additional space to store the mismatched pointer. To sum up, this alternative scores **2 out of 5** in this category.
- **Accuracy**: Searching the Trie Tree that organizes by lexicographical order is more like retrieving a word in the dictionary; if the question is in the saved dataset, it can always return the relative answer. To sum up, this alternative scores **5 out of 5** in this category.

The overall score for AC **4.3 out of 5**, this is determined by the average of the previous scores.

3.3.4 Chosen Approach

Using the desired characteristics mentioned in section 3.3.2, Figure 3 provides each alternative algorithm reviewed in section 3.3.3 with a ranking between 1-5.

Table 3.3.1 Searching Algorithms Rankings

Algorithm	Searching Speed	Space	Accuracy	Average
KMP	2	4	3	3
DFA	3	4	5	4
AC	5	3	5	4.3

According to table 3.3.1, the rank of Searching algorithms, JabberJack will use Aho-Corasick algorithm (AC) as its core algorithm. It is the fastest searching algorithm compared with the other two algorithms. In addition, the highest accuracy of retrieving problems from the saved dataset will give the Chatterjack an efficient way to search and return answers to users.

3.3.5 Proving Feasibility

Aho-Corasick algorithm will support JabberJack with low runtime and high accuracy. Moving forward, the demo of AC algorithm script will get the user's question as input and then retrieve the data from the database and return the response to the problems in a short time. This will show that the algorithm is efficient and can accurately grab data from the database.

3.4 User Interface

The User Interface or UI is one of the most important parts when it comes to any program. In this instance, a UI with a basic input text box and some kind of submit button letting the chatbot know it is ready to analyze the question would be extremely helpful and easy to use as a first iteration. Without an intuitive and well formatted UI, end users could be lost on how to use a program. It is critical that when constructing a UI to think from the perspective of a customer or end user rather than the programmer, because not everyone thinks like an engineer nor are they going to understand the inner workings of the programming.

3.4.1 Desired Characteristics

- **Intuitiveness**: The most important part of a good UI is how intuitive it is to use. The idea behind how intuitive something is how quickly and easily a user can pick up a program. Many programs with good UIs have icons, buttons and formats that are agreed on by the community to be immediately recognizable. An example of this is, there is often a three horizontal line button in the top right of a program or web page that gives several sub-directories of the program.
- **Structure**: It is important to have a strong structural foundation that can prevent users from misusing the program as it was intended in a UI. A proper implementation for this should prevent users exploiting certain elements or conditions. This also means to help users with a way out when they find themselves in a directory they did not intend to be in, Such as a home button.
- **Simplicity**: It is important to have a UI that has meaningful elements on it. One huge mistake for any program is giving its user sensory overload by having buttons and bright colors scattered all over the screen. This is where this element becomes important. There should not be a page full of buttons that all lead in different directions, only elements that provide substance to the program should be included and everything extra should be cut. Minimalism is a huge aesthetic in the market today, Apple being the biggest example of this.
- **Consistency**: Last but not least without consistency in the program the other elements would fall apart as well. It is crucial that when implementing any header navigation bars or adding colors to the background that it is added to the rest of the program to maintain an easy to recognize and use program. All styles and conventions should be consistent with all screen components making it significantly more intuitive as the user explores the program.

3.4.2 Alternatives

There are plenty of other libraries that can be utilized when constructing a user interface, however it is important that the libraries being used include the necessary elements to keep the aforementioned characteristics for a good UI.

- **PyQt**: PyQt is built around the Qt framework, which is a cross platform framework used in a multitude of applications. Using the multiplatform capabilities would future proof the project by ensuring the ability for the chatbot to run on any kind of operating system.
- **Tkinter**: Tkinter is another useful Python library that utilizes visual elements like widgets. Widgets are buttons, check-buttons, labels, canvases, and frames.
- **spaCy**: spaCy uses Natural Language Processing or NLP to integrate nicely with some of python's built-in capabilities. Over the years it has become increasingly popular to analyze data using NLP. NLP is the focal point of this library leaving plenty of interpretation for string based processing which will be extremely helpful on this project.

3.4.3 Analysis

In this section the options will be evaluated based on the desired characteristics listed in section 3.4.1. This section will follow the ranking system laid out in the last paragraph of section 2.0.

- **PyQt**
  This has a wide variety of uses for creating graphical user interfaces, it has an impressive arsenal including QtGui and the QtDesigner modules. These modules provide a plethora of visual elements that can help implement complex systems like drag-and-drop, and dropdown menus.
    - **Consistency**: PyQt is widely accepted as a useful library as well as it's use for multi-platform programming. Utilizing this would help future proof the project giving it a wider reach. In summation, it receives a **4 out of 5** in this category.
    - **Intuitiveness**: PyQt's tools provide a complex system and require a standard of knowledge to use. Although the tools are not all easily able to be picked up, the tools still can be used quite well for testing purposes. In summation, PyQt receives a **2 out of 5** in this category.
    - **Simplicity**: PyQt uses many useful elements that are easy to implement and only needs a bit of excess code to keep simplicity. The general simplicity of the library allows this project to be easily adaptable to the needs of the project. In summation, PyQt receives a **3 out of 5** in this category.
    - **Structure:** PyQt has tools that are well implemented and leave little room for exploitation, but are not structured around it. Due to the tools not being exactly

what is needed of this project it will not be viable enough. In summation, PyQt receives a **3 out of 5** in this category.

The **overall score** for PyQt is **3 out of 5**, this is determined by the average of the previous scores.

- **Tkinter**

  This is extremely useful when it comes to having a wide variety of tools to use when making a well designed UI. Having most of the groundwork laid out for proper UI elements would be helpful in the short term.

  - **Consistency**: Tkinter includes and shows standard use of all the widgets it provides, leaving little room with discrepancy. Since all of these tools can be recognized due to their consistency with other programs it helps users interact more efficiently. In summation, Tkinter receives a **3 out of 5** in this category.
  - **Intuitiveness**: Tkinter's widgets are easy to implement as well as used even for users who are not technically inclined. All of the tools are easy to use and recognize, but they are not entirely needed to meet the requirements of the project. In summation, Tkinter receives a **4 out of 5** in this category.
  - **Simplicity**: Tkinter has a wide variety of widgets but most elements are bland and can easily be too cluttered if not organized correctly. Since this is not an essential part of the project nor is it easy to use it will be better to steer clear from this. In summation, Tkinter receives a **2 out of 5** in this category.
  - **Structure**: Tkinter has little to no security measures implemented for some kind of user authentication. Considering it will be important to have a correct and well implemented user authentication this does not fare well for the requirements of the project. In summation Tkinter receives a **1 out of 5** in this category.

  The **overall score** for Tkinter is **2.5 out of 5**, this is determined by the average of the previous scores.

- **spaCy**

  This is a complex library that uses NLP which is extremely popular with processing any kind of string based data. Using this library it can easily sift through the database and the question given to give a proper response.

  - **Consistency**: spaCy's elements and formatting follows python3 perfectly and is extremely easy to implement the elements across the program with consistency. Using this will keep all formatting consistent as well as efficiently integrate with the separate parts of the program. In summation, spaCy receives a **4 out of 5** in this category.
  - **Intuitiveness**: spaCy's elements are easy to implement and all UI elements can be easily understood by someone who is not technically inclined. This will also

allow for easy translation using NLP so that the user inputted question can be easily deciphered. In summation spaCy receives a **5 out of 5** in this category.

- **Simplicity**: spaCy provides a sleek design for UI elements giving it complexity but the ability to not feel overwhelmed when interacting with the user interface. Due to the simplicity of the library's UI elements, it will be easy to implement an easy to use text box and button that can process NLP efficiently. In summation spaCy receives a **4 out of 5** in this category.
- **Structure**: spaCy has security measurements implemented, none of which are crazy or oriented towards this part but has well structured elements that are hard to manipulate. The elements inside spaCy's library have enough threat mitigation to help meet the requirements of the project. In summation, spaCy receives a **3 out of 5** in this category.

The **overall score** for spaCy **4 out of 5**, this is determined by the average of the previous scores.

3.4.4 Chosen Approach

Using the desired characteristics mentioned in section 3.4.2, table 3.4.1 provides each alternative algorithm reviewed in section 3.4.3 with a ranking between 1-5.

Table 3.4.1 User Interface Rankings

Options:	Consistency	Intuitiveness	Simplicity	Structure	Average
PyQt	4	2	3	3	3
Tkinter	3	4	2	1	2.5
spaCy	4	5	4	3	4

Looking at the averages in table 3.4.1, it makes the most sense to go with spaCy because it meets the most amount of our desired characteristics for the UI end of the program. Not to mention that spaCy also uses NLP which is extremely good for processing string based data and passing it to the back-end which is exactly what is needed for a chatbot.

3.4.5 Proving Feasibility

The first iteration of the program will include UI elements such as a text box, submit button, and other elements to get a working prototype. In the future, it is desired to have no actual physical

interaction between the user and the program. Thus the aforementioned UI elements would probably not make it to the end product; however, they will be critical when it comes to making sure the chatbot is thoroughly able to receive and answer questions correctly and effectively.

4.0 Technological Integration

JabberJack

This section will cover how the chosen solutions of the four major technological challenges will work together to form a comprehensive architecture for the Chatterjack. There are 4 issues addressed in this document and in order to bring them all together Figure 4.1 depicts the expected workflow of the product.

Figure 4.1: Chatterjack Workflow

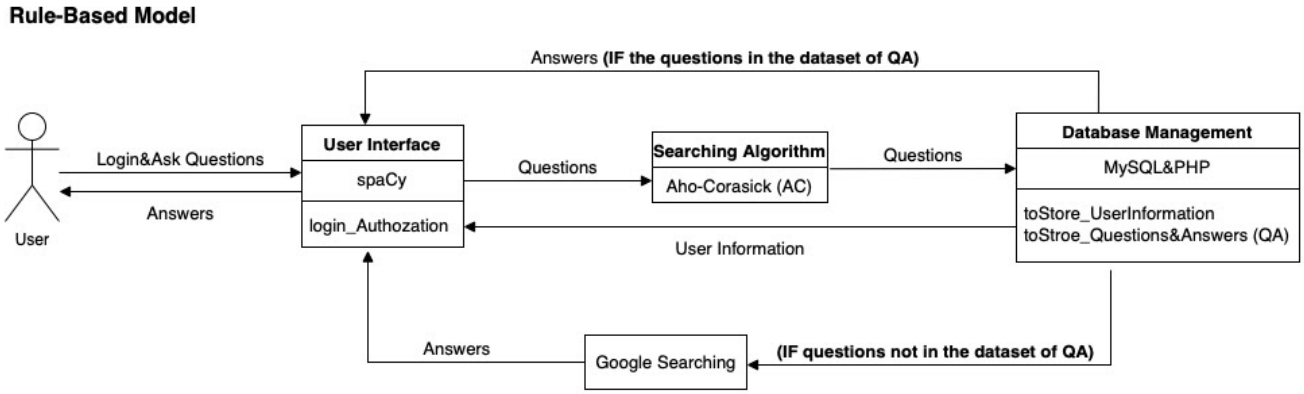

Figure 4.1 depicts the Chatterjack workflow by showing the process that the technologies will go through in order to provide answers to questions. First the user will access the chatbot through the user interface that will utilize spaCy. Using this user interface the user can ask the chatbot questions through text. The chatbot will then utilize the algorithm to search through the database of preconceived questions and answers. The Chatterjack chatbot is a Rule-Based Model that uses a saved dataset of Questions and Answers (QA) Each solution will integrate carefully with one another to create a fully functional system [22]. A rule based model chatbot is a chatbot that utilizes a database of preconceived questions and answers and follows a set of rules that guide how the chatbot answers questions. This database will be managed using MySQL coupled with PHP. Once the algorithm finds the answer that corresponds to the question asked it will pass it off to the user interface. If the algorithm does not find the answer the chatbot will search the internet for the answer and hand what it finds off to the user. The user interface will then display the answer appropriately back to the user. For questions the chatbot does not have pre-answered it will scrub data from the internet and add it to the database so that the question can be answered in the future.

In addition to this workflow there is a second underlying workflow for administrators that will manage the chatbot. Administrators will be able to login to the chatbot and will be able to manage the database that the chatbot grabs from. Administrators will use the user interface to access the database and are able to add or delete question answer pairs. This allows the chatbot to add new questions manually as well as automatically through the web scraping. With all of these technologies working together we hope to deliver a chatbot that will efficiently and effectively answer questions about NAU in a timely manner.

5.0 Conclusion

JabberJack

Information surrounding Northern Arizona University is scattered across the internet on several different websites and portals. This makes it difficult for students, parents, and faculty alike to find answers to questions that they have. A lack of readily available information often leads to both frustration and confusion as users have to manually filter through several different sites to find answers. Most users end up emailing or contacting human resources in order to find the information that they need. This takes people away from other tasks that may be more important than answering a question. The Chatterjack chatbot is the solution to this; it can both provide answers to questions and allow NAU employees to avoid answering tedious questions.

Table 5.1: Chosen Technologies

Technological Challenge	Chosen Solution
Authentication and Security	PHP
Database Management System	MySQL
Search Algorithm	Aho-Corasick algorithm (AC)
User Interface	spaCy

In table 5.1 the chosen technologies to solve challenges are what will be implemented to create a functional chatbot. This document analyzed the options that are available for authentication and security, database management systems, algorithms, and user interface packages. After close analysis of several different alternatives for each of the challenges the project will utilize PHP to enforce security and to provide user authentication. MySQL will be used as the database management system in order to store and retrieve the questions and answers. The chatbot will utilize the Aho-Corasick algorithm in order to retrieve data from the database efficiently. The user interface will utilize spaCy, a python package specifically designed to be used in conjunction with chatbots, for the user interface elements.

The in depth analysis of each technology has provided the best possible solution for each of the challenges outlined in the document. With all of these technologies working together, team JabberJack will be able to create a functional chatbot that can answer questions efficiently. This

project will set the foundation for more complex implementations of the Chatterjack in the future.

6.0 References

JabberJack

Introduction

[1] Miklosik, Andrej & Evans, Nina & Qureshi, Athar. (2021). The Use of Chatbots in Digital Business Transformation: A Systematic Literature Review. IEEE Access. 9. 106530-106539. 10.1109/ACCESS.2021.3100885.

[2] NAU. 2021.(2021). Retrieved November 3, 2021 from https://in.nau.edu/campus-health-services/

[3] NAU. 2021. Facts and stats. (2021). Retrieved November 3, 2021 from https://nau.edu/about/facts-and-stats/

[4] Rooein, Donya. (2019). Data-Driven Edu Chatbots. 46-49. 10.1145/3308560.3314191.

Authentication and security

[5] Quora. 2018. Should I learn PHP in 2019? is it still worth it? - quora. (2018). Retrieved November 3, 2021 from https://www.quora.com/Should-I-learn-PHP-in-2019-Is-it-still-worth-it

[6] Tutorialspoint. 2021. PHP Login Example. (2021). Retrieved November 3, 2021 from https://www.tutorialspoint.com/php/php_login_example.htm

[7] Cloudflare. What is a WAF? | web application firewall ... - cloudflare. Retrieved November 3, 2021 from https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/

[8] W3Schools. 2021.(2021). Retrieved November 3, 2021 from https://www.w3schools.com/PHP/php_intro.asp

[9] Flavio Copes. 2021. JWT authentication: Best practices and when to use it. (July 2021). Retrieved November 3, 2021 from https://blog.logrocket.com/jwt-authentication-best-practices/#brief-introduction-to-jwt

[10] Paul Jackson. 2021. What is php? write your First PHP program. (October 2021). Retrieved November 3, 2021 from https://www.guru99.com/what-is-php-first-php-program.html

Database Management System

[11] Mark Smallcombe. 2020. *PostgreSQL vs MySQL: The Critical Differences* (May 2020). Retrieved November 3, 2021 from https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/

[12] Diederich Tom. 2018. *5 Limitations of MySQL with Big Data* (January 2018). Retrieved November 3, 2021 from https://www.gridgain.com/resources/blog/5-limitations-mysql-big-data

[13] LDB Team. 2019. *PostgreSQL vs. MySQL: A 360-degree Comparison [Syntax, Performance, Scalability and Features]* (December 2019). Retrieved November 3, 2021 from https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features

[14] Wikimedia Foundation. (2021, October 31). *SQLite*. Wikipedia. (July 2021). Retrieved November 3, 2021 from https://en.wikipedia.org/wiki/SQLite

Algorithms

[15] Wikimedia Foundation. (2021, October 26). *Knuth–Morris–Pratt algorithm*. Wikipedia. (July 2021). Retrieved November 3, 2021 from https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm

[16] Wikimedia Foundation. (2021, October 26). *Deterministic finite automaton*. Wikipedia. (July 2021). Retrieved November 3, 2021 from https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm

[17] Wikimedia Foundation. (2021, October 26). *Aho–Corasick algorithm*. Wikipedia. (July 2021). Retrieved November 3, 2021 from https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm

User Interface

[18] *Spacy · industrial-strength natural language processing in python*. Industrial-strength Natural Language Processing in Python. (n.d.). Retrieved November 3, 2021, from https://spacy.io/

[19] Wikimedia Foundation. (2021, October 26). *Spacy*. Wikipedia. Retrieved November 3, 2021, from https://en.wikipedia.org/wiki/SpaCy

[20] *What is Pyqt? Riverbank Computing | Introduction. (n.d.).* Retrieved November 3, 2021, from https://riverbankcomputing.com/software/pyqt/intro

[21] Maze. (2021, September 2). *What is user interface design?* Maze. Retrieved November 3, 2021, from https://maze.co/collections/ux-ui-design/ui-design/

Technological Integration

[22] Stefan Kojouharov. 2016. *Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot* (2016). Retrieved November 3, 2021 from https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c