

# Final Report

## IntelligentChild

## TAMU School of Nursing

Team Member Names :

Emmeline Blevins

David McDonald

Wyatt Masterson

Samyam Pandey

Prakhar Singh

CSCE 482 – Senior Capstone Design  
Spring 2023

Texas A&M University

Department of Computer Science and Engineering

# 1. Abstract

In the modern era of overwhelming information, searching for trustworthy resources on specific topics can prove to be a daunting task. The traditional search engines present numerous search results that may not be relevant to the user's query, thereby posing a challenge in identifying authentic sources. However, we have developed a solution that caters to this challenge. Our project aims to create a curated search engine designed explicitly to provide users with high-quality resources on maternal healthcare and pregnancy-related topics. Our advanced algorithms gather and classify relevant information from dependable sources, ensuring the availability of accurate and reliable resources. Our search engine is enhanced with natural language processing capabilities to provide users with a personalized and user-friendly experience. Our platform caters to a diverse audience, including expectant mothers and users with a keen interest in pregnancy-related topics. Our commitment is to offer valuable insights and guidance on these subjects within a comprehensive and convenient resource. In summary, our innovative approach to search engines prioritizes quality, accuracy, and user-friendliness - offering the necessary confidence required to navigate the intricate realm of information with ease. To ensure the accuracy and effectiveness of our search engine, we adopted a two-fold validation approach. Firstly, we created a validation set from our training data set to test the search engine's performance in identifying and presenting relevant resources. Secondly, we conducted user acceptance testing, where users tested the search engine and provided feedback on its accuracy and usefulness. We used this feedback to further refine and optimize the search engine's performance. In evaluating the search engine, we utilized a validation set along with feedback from users who submitted their own queries to assess the accuracy and relevance of the search results. This allowed us to gauge the search engine's performance in identifying and presenting useful resources to users. Through this iterative process, we were able to improve the search engine's functionality and optimize its performance. Overall, our rigorous validation process and user-centric approach allowed us to develop a reliable and efficient search engine that caters to the diverse needs of its users.

## 2. Introduction

The primary objective of this project is to develop a highly efficient search engine tailored to provide users with high-quality resources on maternal healthcare and pregnancy-related topics. The search engine will utilize advanced algorithms to gather and classify relevant information from credible sources, ensuring the availability of accurate and reliable resources. To enhance the user experience, our search engine will be equipped with natural language processing capabilities, providing users with a personalized and user-friendly interface. The overarching goal of this application is to provide a medical search engine that is accessible to people without a medical background, facilitating easy comprehension of the

resources presented. We will evaluate the performance of the application by first verifying the accuracy of the search engine's output against the provided testing data. We will then utilize user acceptance testing (UAT) to assess the application's effectiveness in meeting the needs of its target audience. Our commitment to the success of this project is unwavering, and we aim to develop a search engine that will significantly improve access to maternal healthcare and pregnancy-related information.

### 3. Related Work

Summary: [MedSearch: a specialized search engine for medical information retrieval](#)

#### Overview

MedSearch was created to address the issue of existing Web search engines not providing useful medical information, whether because of not interpreting users' queries correctly or not providing sufficiently diverse results. MedSearch is able to accept incredibly long search queries and extract keywords from them, as well as find results that are not overly similar to each other. It can also suggest relevant medical keywords to users, to help them refine their searches. When evaluating MedSearch, the research team compared the usefulness of MedSearch's results to those of its top competitors at the time, Google Health and Healthline. The team concluded that MedSearch's unique query processing enables it to deliver more relevant results than either of these competitors.

#### Discussion

I liked MedSearch's method of consolidating lengthy queries into more useful mid-length queries, and I think something similar is very likely to be useful when building the Intelligent Child. I also liked that it only crawled a set of websites considered trustworthy, from which to serve results; this is similar to the database system Intelligent Child will use. I felt that the functionality of delivering sites with significantly differing results was a good way to help people with low levels of medical knowledge find less-common results for their symptoms or situations. However, I disliked how unclear the paper was about exactly how the usefulness scores for MedSearch's, Google Health's, and Healthline's top results were obtained. I felt that I would have been better able to trust these results had I been given more information about what demographics were interviewed, how participants were divided between the three search engines, and various other procedural details.

---

Summary: [EMERSE: Electronic Medical Record Search Engine](#)

## **Overview**

Emerse is a powerful search engine for free-text documents in the electronic medical record. Emerse was built in order to address the need for searching the medical record for research and data abstraction. Emerse is easy to use by being an intuitive user interface. In addition to having a easy-to use interface that it is easy for users that have minimal computer experience Emerse is able to create complex search queries. When evaluating Emerse it was concluded that Emerse is ideal for retrospective chart reviews and data abstraction and may have potential for clinical care as well.

## **Discussion**

I liked how Emerse was first introduced 6 months ago and during that time over 90 users have been registered and searches have been conducted on over 5800 unique patients in their health system. This shows how powerful/useful Emerse is and how later on time it is going to be super powerful. In addition, I liked how Emerse just in 6 months not just got multiple users but also have gotten overwhelmingly positive responses. The one thing I disliked is that the search engine isn't efficient or as accurate as it can be but this is due to it just being in its beta stage. It is related to our paper because of the use of natural language processing used in Emerse which will be contributed to Intelligent Child.

---

Summary: <https://slack.engineering/search-at-slack/>

## **Overview:**

The web application Slack allows users to search for keywords in the channels they belong to and from their previous conversations. They used to only sort the results by recent, but decided to implement a relevant sorting algorithm for the search results as well. The relevant search takes into account the Lucerne score of the document - how well it matches the query terms (they use a third-party search engine called Solr for this). Unfortunately, the Lucerne score alone wasn't enough, so they decided to rank the results from Solr using a custom model. The model labels results that were clicked more frequently, and gives them higher priority in the search results. Additionally, the model is trained on various attributes such as the age of the method, lucerne score, whether the message had reactions, among others. This allowed the results to be as tailored to the individual user as possible

## **Discussion:**

Out of the hours of research I conducted for this assignment this seems to be the closest example of what we are looking for. I can imagine us labeling the data with more custom tags for our model to predict even finer, but the overall idea of using a "vanilla" semantic search algorithm, then building a secondary ranking on top of that, is very, very likely how we are going to be implementing the search in our application.

---

Summary: [IBRI-CASANTO: Ontology-based semantic search engine](#)

**Overview:**

IBRI-CASANTO is an Ontological search engine for Colleges of Applied Sciences, Oman. The main focus is to transform user queries that may not have the necessary keywords or have typos into understandable, actionable queries that can plug right into the database and retrieve accurate results. The first thing needed is a graph describing the relationship between various classes and individuals that exist in the database. For example, a college database may have a Classes node in the graph, that connects to both Students and Professors. This allows for much easier labeling of data and paints a clear picture of how all of the data relates to itself. Now, when users query the application, their query is reformulated using ontology and processed via two different search methods, both of whose results are compared and the most common one returned. The search methods are keyword search via Lucerne (same as the one slack uses) and semantic search via Apache Jena Fuseki. The end result is that the user is presented with 3 methods of searching. Keyword search, a list of predefined queries, and a final method where the user can come up with their own query.

**Discussion:**

Honestly, this isn't exactly what we are looking for but the concept of the ontological graph to represent the data could be quite useful for our purposes. Additionally, the combination of semantic and keyword search is a novel idea we can look into while building this application.

---

Summary: [Analysis of the Combination of Natural Language Processing and Search Engine Technology](#)

**Overview:**

When a user is faced with millions of web pages and sites, they can only attempt to access the relevant ones by describing, in their own words, what they are searching for. NLP-Search Engines will fully replace traditional search engines once advanced enough as they more accurately reflect the needs of users. Natural language processing can be grouped into three aspects - formalizing linguistics into mathematics, the mathematical formalization of linguistics being used in algorithms, and the process of the NLP programs implementing these algorithms. Search engines are grouped into four different categories - directory engines (phonebooks), robot search engines (google page rank), meta search engines (combinations of other engines), and specialized search engines (medical record database search). There are a few key issues of NLP, one of which being word segmentation and ambiguity. It is difficult for a computer to correctly determine the context and exact influence of words on one another within a sentence. Another

issue is semantic processing, which is how a computer breaks down and understands the sentence structure of a query.

**Discussion:**

I liked that this paper gave a solid, top-level overview of some of the challenges and solutions to NLP search engines as they currently exist. While this paper won't be directly relevant to the project in terms of specific implementations or theory, this is a great resource for a "starting point" in terms of understanding these two intertwined technologies. I did not like how short this paper was, seeing how there may have been room to expand more on some of the specific subcategories of NLP or search engines. This paper is slightly less relevant to the project as it references the difficulties of converting the Chinese language into usable computer data - however, because this is not explored in detail, the relevance when compared to the English language is still high. Overall this was a great resource to grasp a better understanding of the technologies we will soon be working with.

---

**Summary:**

<https://www.sciencedirect.com/science/article/pii/B9780128153680000026?via%3Dihub>

**Overview:**

Due to an exponential increase in the volume of healthcare data, relational databases are proving to be unfit for the larger numbers of users and data. Because of this, there is a need for a faster and more efficient database. Using a NoSQL database instead allows for an increase in performance and availability. Although there may be some challenges with data migration, it is proven to be much better than using a conventional relational database.

**Discussion:**

I find this article interesting and I do very much agree. In my time at university, I have come to learn that the recent emergence of NoSQL databases have proven to be useful, and can improve upon database design significantly.

---

**Summary:**

<https://www.sciencedirect.com/science/article/abs/pii/S1386505613000166>

**Overview:**

FindZebra designed as an evaluation approach for web search engines for rare disease. FindZebra includes 56 real life diagnostic cases, performance measures, and etc.. FindZebra is powered by open source search technology and uses curated freely available online medical information. Findzebra outperforms google search. They outperform Google search in both default set-up and customized to the resources used by FindZebra.

### **Discussion:**

I find this article interesting and I do very much agree. This article gave us motivation for a search engine that can be better than google search because google search is the top search engine. I also like this article because it demonstrated different ways to implement a search engine for our own project. Furthermore, I liked this article because it gave us an idea of the challenges we might face when trying to implement a search engine.

---

**Summary:** [SQLizer: query synthesis from natural language](#)

### **Overview:**

This paper was a research study done at UT Austin where the researchers developed a new method of obtaining SQL queries from natural language searches. This is done in three distinct steps. First, the query generates a “sketch” using semantic parsing, meaning a “query skeleton” is created to represent the shape and form of the query. This portion of the process simply splits a search into its semantic components. Second, a type-directed query synthesizer uses the previous sketch to generate a list of possible queries, comparing to the database schema and contents to assign the different queries a confidence score. Third, the sketch is “repaired”. It is very common for there to not be any high confidence queries, or for there to be too many high confidence queries. A database of repair techniques is applied to the sketch in order to generate more confident searches. Interestingly, this entire process is *database agnostic*, meaning that the first sketch is generated purely off of the NL presented, then uses the existing database to check its own generated queries. The query generation does not look at the database; however, the query refinement does.

### **Discussion:**

This paper seems to be both very helpful and relevant to this project. A process for a database agnostic SQL query generator based on NL input is exactly the problem we are looking to solve. This paper also provides a solid overview of implementation and practical applications, giving sufficient algorithm pseudocode to make this implementation in our project feasible. Overall, this paper significantly increased my technical understanding and knowledge of the problem at hand, as well as provided a relevant and achievable solution. I will be sharing this paper with all of my group members since it does seem to be one of the most holistic evaluations of the problem we have set out to solve.

---

Summary: [Development of Mobile-Based Personal Health Record Maternal and Child Using Glide App: Development and Usability Study](#)

**Overview:**

In response to issues of malnutrition among children, the elderly, and pregnant and lactating mothers, this team designed an application to aggregate medical records from these demographics. They gathered stakeholders who worked in first-level medical care and had a deep knowledge of the relevant issues. After gathering requirements, they designed the necessary database to serve their project's needs. Subsequently, they designed a prototype graphical user interface and began the first round of user testing, using the System Usability Scale method. Overall, they found that the usability of their resulting application was significantly better than average.

**Discussion:**

This application differs greatly from ours in that it is aimed toward medical professionals, and the mothers whose information it aggregates do not have access to it. It also uses the waterfall method, unlike our decision to use the agile development process. However, the methods used to test the resulting application seem like they would be appropriate for any application. When our team reaches the user testing phase of our project, we can consider using the System Usability Scale to help direct our users' thoughts. The numerical score that results from using this scale will also help give our results a greater feeling of objectivity.

---

Summary: <https://www.hindawi.com/journals/jhe/2020/8839524/>

**Overview:**

A chatbot health system based on fuzzy logic rules and fuzzy inference was created and used to assess symptoms of diseases in Nigeria. It uses a knowledge base consisting of known facts about diseases and symptoms. Based on the input, a fuzzy support vector machine is used to predict the disease. The inputs are then recognized with natural language processing, and then forwarded to a human doctor for decision support. Finally, a notification is sent to the user with a message displaying the end of the diagnosis process.

**Discussion:**

Although this application is slightly different from ours with the fact that it has to do with medical disease diagnosis and not maternal health, it is important to keep this in mind due to the



fact that it uses natural language processing. On top of that, we will be using a knowledge base with known facts as well. Since the structure is fairly similar to what we want to do, it would be good to study how this system works and try and emulate some of its principles.

## 4. Engineering Standards

### 4.1 Environmental and Health/Safety Concerns

Health Concerns:

- Correctness
  - When using the search engine, there is a possibility that it may give the user incorrect information about their medical situation. This can have a negative effect on the user's well-being.
- Availability
  - The search engine and database systems must always be available, since users may urgently need to seek help about their medical situation. If the services aren't available, this can negatively affect the users' well-being.

Environmental Concerns:

- Waste production
  - Because this project is entirely software based, there are not any physical production/manufacturing concerns based on producing any products. The main waste produced would be carbon emissions from the electricity used to run the servers hosting this application.

### 4.2 Social, Political, and Ethical Concerns

Social Concerns:

- Real impact of maternal health outcomes
  - This project, which will recommend healthcare providers and resources, by definition will impact local health outcomes. As such, ensuring that the software is accurate, reliable, and secure will have a direct impact on the health and wellbeing of communities being served.

Political Concerns:

- Recommendations of maternal/reproductive health.

- Since this project deals with a software engine and database for recommending maternal health resources, the political implications of where this is deployed are enormous. Since the customer has noted that this may be deployed on a state-by-state basis, each state will effectively have even more control over the visible and reachable maternal health resources. This could mean women being sent to pregnancy crisis centers instead of abortion clinics or vice versa, certain healthcare providers being “blacklisted” by a state who doesn’t want to promote them, or various other political concerns.

#### Ethical Concerns:

- Personal Privacy
  - If users’ personal information, or even the content of their queries, is leaked, this would be a major violation of privacy. This could also violate HIPPA or another type of healthcare information privacy act.
- No Censorship
  - We must ensure that all users have access to the most accurate health information and access to medical services.
- Scraping Data
  - We should always ensure that we only gather data from users to support features that benefit them.
  - When using data scraped from the internet, we must be sure to independently confirm its validity and accuracy before presenting it to users.

## 4.3 Manufacturability, Sustainability, and Economics

#### Manufacturing Concerns:

- None
  - This is a software project and will produce no manufacturable physical items. Therefore, manufacturing costs and accommodations need not be considered.

#### Sustainability Concerns:

- Scalability
  - The customers have discussed this project being implemented on a state or even nation-wide level. This means that we must consider that, although originally only being implemented with respect to Brazos Valley healthcare providers, the system must be robust and sustainable enough to handle a significantly larger volume of traffic and a much larger database to search over.

- Ecological Footprint
  - As a digital, cloud-based service, this project has a low physical footprint. However, when choosing cloud computing services for the future, it will be critical to balance considerations of speed and availability with the environmental policies of each company under consideration. A company with slightly lower speed that invests in carbon offsets may be more desirable than the company with the fastest servers on the market.

#### Economic Concerns:

- Scaling Costs
  - Since this project has yet to be implemented, we don't yet know if we will include any third party software libraries. Many of these operate on a "pay if your product makes money" model to encourage developers to work with them. In this case, we would need to be transparent and proactive with communicating to our customer about these possible future costs (especially since ignoring them could have legal ramifications).
- Funding
  - Our client has proposed multiple avenues by which the project could obtain future funding, such as the sale of branded merchandise or the option to offer a premium subscription service.

## 5. Requirements

---

### 5.1 Overview

The IntelligentChild project seeks to develop an intelligent database search system for maternal healthcare resources based on natural language inputs returning results from a curated list of vetted and trusted providers. As it is very difficult for young, inexperienced mothers to acquire help and resources, this application is attempting to solve this problem by making a search for resources easy and intuitive. Ideally, an expectant mother should be able to open the IntelligentChild application, type in a search in their own language and verbiage, and have a list of trusted providers sent back to them.

### 5.2 User Stories or Usage Scenarios

1. As an expectant mother, I should be able to search for resources in my own language because I am not familiar with medical terminology.
2. As an expectant mother, I should only receive relevant resources from my search because I do not have time to parse out unwanted results.
3. As a system administrator, I should be able to add resources to the IntelligentChild database in case new resources become available.
4. As a system administrator, I should be able to edit resources in the IntelligentChild database in case existing resources deprecate, move, or otherwise change in a meaningful way.
5. As an expectant mother, I should only see a handful of curated resources because too many may overwhelm or confuse me.
6. As an expectant mother, I should be able to search/browse the application knowing that data is secure because the data being accessed can be quite sensitive and I may not want others knowing what I am searching.
7. As an expectant mother, I should be able to receive the information I need without it being unavailable, in case the information needed is critical.
8. As a system administrator, I should be able to identify if any existing resources are irrelevant and should be removed.
9. As a young expectant mother, I should be able to understand how the application works easily.
10. As a system administrator, I should be able to change the interface to make the application easier to use.
11. As an expectant mother, I should be able to search for financial aid for which I am eligible, because it would waste time to apply for resources for which I am not eligible.
12. As a system administrator, I should be able to add fields to resources in the database in case the medical landscape changes in a way that makes additional information necessary.
13. As an OB/GYN, nurse, midwife, or other medical professional, I should be able to refer my clients to the application, in case my client has a question when I am unavailable.
14. As an OB/GYN, nurse, midwife, or other medical professional, I should be able to request that my information, or my practice's information, be added to the database, so that expectant mothers in need can be more effectively directed to the resources they need.
15. As an OB/GYN, nurse, midwife, or other medical professional, I should be able to provide medical information from the resources the application provides to my client.
16. As a medical professional, I should be able to verify that the information the application provides to my client is factually correct and should be followed.
17. As a system administrator, I should be able to identify any searches that need to be fixed and extended further
18. As a system administrator, I should be able to encrypt and protect the data of all users in the IntelligentChild database/application.
19. As a provider, I want to ensure that the information provided to expectant mothers is only the most important and necessary information
20. As a young and expectant mother, the searches need to be executed quickly and not take very long

### 5.2.1 Definition of Success

1. Users should be able to put a query in with natural language and be returned a resource if a relevant one exists
2. Queries with no connection to any resource will not have any resources returned.
3. Administrators can add a new resource and have it be accessed by others on the site.
4. Administrators can edit a resource and have the changes reflected on the site.
5. Queries return 0-5 results based on the relevancy of resources in the database.
6. Application passes automatic security tests and is subjected to/passes heavy testing on data security.
7. Site should not be down more than 2% of time deployed
8. There should be an interface to provide data to train the model on
9. The application is deemed “easy to use” by at least 80% of users rating the application at least an 8 out of 10 on a Likert scale questioning if the website was intuitive and easily understandable.
10. System administrators should have an interface to change the look of the website
11. During UAT, user questions about the relevancy of resources returned when asking about financial aid should have at least a 75% positive response rate.
12. System administrators should be able to add, edit, or change resource fields within the database.
13. Users with a medical background should average at least an 8.0/10 on a likert scale asking the trustworthiness of a website
14. System administrators should be able to add specific resources into the database
15. The information returned by the application should be relevant to the query as per UAT relevancy score results having at least a 7.0/10 average on a likert scale.
16. The providers should be able to ensure the data returned concerning them is accurate and up to date
17. Based on free response UAT questions, the system administrator should be able to identify queries that should have had responses but didn’t and make adjustments to the model from that
18. The database should be secure and preferably encrypted
19. The system should only return what is necessary for the user making the search to take action, and nothing more
20. Queries should ideally execute in under 1 second

### 5.3 Requirements Models

Weights add to 1 for each sub-category (i.e. All objectives in “Online/Easily Accessible” sum to 1)

<b>Online/Easily Accessible</b>	<b>Appealing Visual Design</b>	<b>Easily usable without search engine skills</b>	<b>Provide relevant and important information</b>
<b>0.3</b>	<b>0.1</b>	<b>0.3</b>	<b>0.3</b>

High Availability (0.3)	Intuitive Design (0.3)	Natural language processing (0.5)	Database with medical information (0.5)
Mobile browser compatible (0.5)	Matching thematically with the rest of the OliviaHealth suite (0.5)	Processes language with slang, misspelling, etc... (0.5)	Does not provide erroneous information (user may not be able to tell) (0.35)
Screen reader compatible (0.1)	Appealing to target demographic (young first time mothers) (0.2)		Prompts user to enter information absent in initial query that is necessary to provide best results (0.15)
Search engine optimized to appear high in Google rankings (0.1)			

## 5.4 Prototypes

We are pleased to inform you that our prototype encompasses a rudimentary website interface that prompts the user to submit an inquiry regarding any topic of their choosing. Subsequently, our Intelligent child algorithm will process the query and return a relevant response aligned with the user's initial request. Please find below a depiction of the basic application prototype for your perusal.

Intelligent Child

Search the Intelligent Child database answers to your questions!

Results

## 5.5 Pilot Studies

In light of our initial phase of pilot studies, we conducted the first round of user testing and observed that 67% of participants acknowledged the resources provided by our system as

pertinent and valuable. Furthermore, 86% of users expressed confidence in the credibility of the returned resources, while 48% of participants favored our system over other search engines such as Google. With the intent of gaining more insight into the user experience, we plan to conduct additional studies and provide further information.

## 6. Design

### 6.1 Overview

It has been determined that a NoSQL database will be necessary to host the data provided by the Olivia Health team for the Intelligent Child project to function correctly. The Olivia Health team has provided guidelines which dictate that this database must be hosted by STAR VM, while the software deployment will be handled by our TCAT sysadmin. Additionally, it may be necessary to utilize Python scripts to process the CSV files provided by the Olivia Health team and insert the data into our database. As of now to host locally the database is hosted on MongoDB and the web application is hosted on pythonanywhere.

Our team has chosen to work on the project using an Agile methodology, with the initial focus being on developing a minimum viable product before adding additional functionality to reach the final goal. The specifics of the data set, examples of representative questions that the search engine should be able to answer, and the profile of the typical end user have yet to be received from the Olivia Health team. However, upon receipt of this information, we will be able to select a suitable search engine algorithm to query the provided database.

As a software-based project, the Intelligent Child will not require any significant hardware acquisitions. All team members already possess reasonably powerful personal computers, which are necessary for enrollment in TAMU's Computer Science program and will be utilized for creating and testing the product. Additionally, the finished product will require web hosting, which the Olivia Health team has determined will be hosted on STAR VM and managed by our TCAT sysadmin.

### 6.2 Comparison of Potential Solutions

Because the problem presented by Olivia Health is quite narrow, there are not any existing one-to-one solutions that implement exactly what we are trying to achieve. However, there are a plethora of software solutions to natural language processing, intelligent search, and generation of search queries from natural language. Some of the relevant and existing solutions are outlined below, along with a short discussion about their relevance to the IntelligentChild problem.

- SQLizer: query synthesis from natural language
  - This solution comes from a paper published in the Association for Computer Machinery journal in 2017 by a group of UT researchers. A model for a

database-agnostic SQL query generator was created and has outperformed many similar models in terms of accuracy and efficiency.

- This model more or less solves one of the main technical hurdles of this project. However, there are no free or open source implementations available.
- However, this solution has two main issues related to the OliviaHealth project:
  - This specific implementation is very computationally heavy. Since we have server limits as dictated by the AWS license, a solution like this is not feasible.
  - This implementation is incredibly complex and requires existing expertise in machine learning, models, and natural language processing. Our project must have an MVP by the end of the semester, so this solution is not optimal.
- MedSearch: a specialized search engine for medical information retrieval
  - This solution comes from a 2008 paper published in the ACM journal noting that one of the main problems in medical searches is a user's lack of knowledge about the search they are performing. This paper has a lot of great insights with how this solution handles NLP in a medical context.
    - This paper specifically tackles generating medical queries from natural user inputs. Notably, users often over explain their conditions in plain english, use incorrect terminology, or mistype their questions.
      - By shortening the often inflated queries, searches can be performed much more quickly with this bit of preprocessing.
      - This implementation's focus on efficiency may help us with our limited resources hosted in the AWS cloud server.
  - The main problem with this implementation is how dated the paper is.
    - Written in 2008, this paper focused more on the technological and computational limitations of processing long medical queries and sought to make these types of searches more efficient. 15 years later, the computational hurdles are much smaller.

There are other related solutions, however the two listed above were the most relevant. Some of the other systems we looked through while designing our application were:

- EMERSE: Electronic Medical Record Search Engine
- Slack keyword search relevance algorithm
- IBRI-CASANTO: Ontology-based semantic search engine

Most of the implementations we studied were simply too far off from our intended solution; however, they did provide a good idea of possible challenges and hurdles we should be considering. The main concerns for this project, as revealed by these related works, will be:

- Computational resource management



- Accuracy and confidence in generating search queries
- Efficiency of search execution
- Differentiating medical searches from other searches (A user will construct a question very differently if trying to locate a restaurant, YouTube video, Amazon item...)

## 6.3 Data Design

We decided to use the NoSQL database MongoDB for our current implementation of the project. This is for 2 reasons:

1. MongoDB is very easy to run locally and can also plug into many hosting solutions
2. MongoDB is a non-relational database, which we need for our project

At present, the data provided for the Intelligent Child project consists solely of a CSV file that contains a list of resources and providers, each with numerous fields that provide details such as their name, address, description, and other distinguishing features. Currently, these providers and resources do not have any inherent relation to one another, and there may be fields that are missing due to data entry errors or other reasons. Additionally, some fields may need to be added or removed, and there are other categories of data that will need to be added at a later stage, which do not conform to the format of a resource or a provider.

Furthermore, our team manually researched resources in the Bryan area that could be added to our database and curated a small list. We had to gather all of the other necessary information such as address, phone number, and a description for each of these new resources. This allowed us to have a much more robust list of resources for testing in our first few stages.

Because of this we went with MongoDB, a non-relational database that stores documents in collections. The advantages of this over a traditional SQL database are:

1. Items added to a collection in the database don't need to follow a specific schema or structure (unless specified). So fields can be missing or added in new documents and it will still work (*Databases and Collections*)
2. Non-relational databases allow for complex fields in an object structure, which is useful for complex data like we are working with
3. When a query is performed only one collection (table) needs to be checked, as opposed to SQL databases where you may have to go through 3 relational tables in a single query

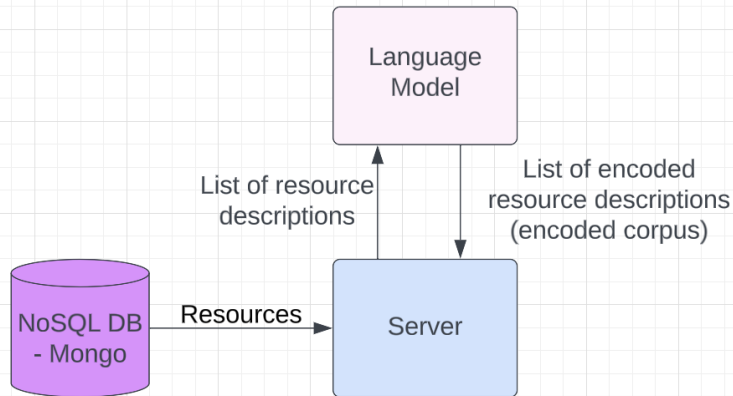
```
_id: ObjectId('6420a681dd7ba3a6f34f8606')
Last Name: "City of College Station Community Development"
First Name: "City of College Station Community Development"
Title: "Community Resource"
Organization: "City of College Station Community Development"
Email: ""
Cell Phone: ""
Work Phone: "979-764-3778"
Role Rel'ps: ""
Description: "Down payment housing assistance and deposit assistance. The City of Co..."
TAMU Affiliation: "No"
Street Address: "1101 Texas Ave."
City: "College Station"
State: "TX"
Country: "United States"
ZIP: 77840
Days of Operations: "M-F"
Provider (M/F): ""
Community Resource/ State Resource: "Community Resource"
PDescription: "City of College Station Community Development-City of College Station ..."
```

```
_id: ObjectId('6420a681dd7ba3a6f34f860d')
Last Name: "Pride Community Center"
First Name: "Pride Community Center"
Title: "Community Resource"
Organization: "Pride Community Center"
Email: ""
Cell Phone: ""
Work Phone: "979-217-1324"
Role Rel'ps: ""
Description: "Services for LGBTQ+
              Provides a safe place for persons of all sexual o..."
TAMU Affiliation: "No"
Street Address: ""
City: ""
State: ""
```

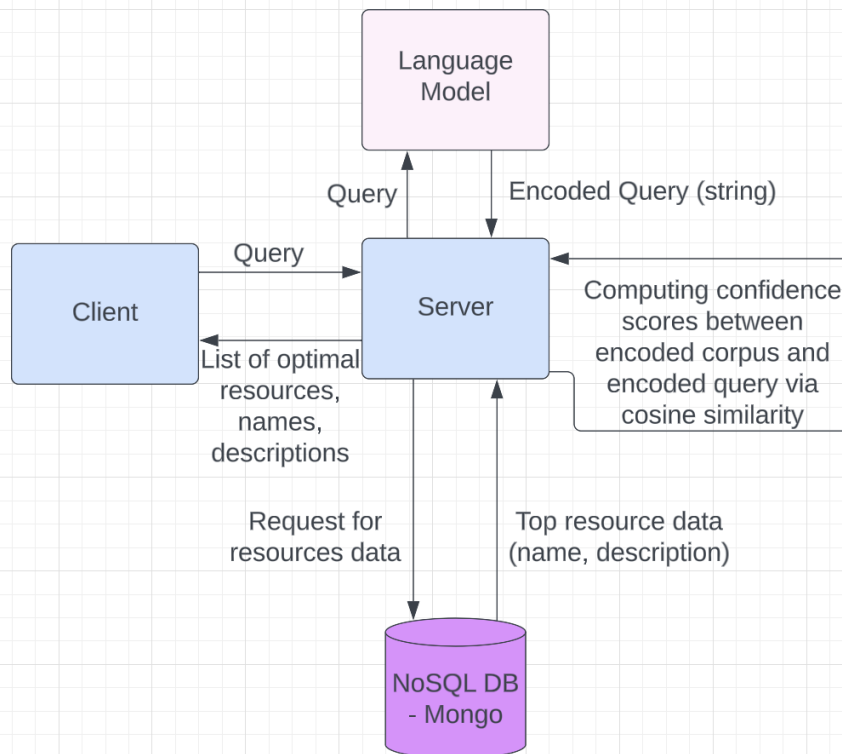
Above is a screenshot of our current database.

## 6.4 Functional Design

### Process for Encoding the Corpus



### Process for Running a Query



When a user enters the IntelligentChild ecosystem, they will be searching for resources related to medical questions they have. These users will not have a large medical knowledgebase, meaning that their “plain english” searches must be refined and understood by the application. This means that their natural language must first be encoded before being put into a search

engine to look over a database of curated and trusted sources. The relevant sources will then be returned to the user.

The minimum viable product for this application will consist of four distinct components, each serving a specific purpose within the program. These four components are the user landing page, natural language processing, search engine, and database. One possible stretch goal for this application would be a context-based search (e.x. more information is asked from the user, the previous searches are considered), however the MVP does not include this functionality.

The first diagram above demonstrates the encoding of the corpus. This occurs before any user accesses the site, and is only executed once: when the site is first initialized. This allows the user's encoded query to have a list of encoded resources to compare the query to. The second diagram is a basic functional flow of the program. The user gets on the landing page, types in a query, the query is encoded, the encoded query is compared to the encoded corpus and the top k results are returned. Then, the results with a confidence score over a certain threshold (0.3) are displayed to the user in the results page.

## **6.4.1 Module Specifications**

### **6.4.1.1 Client**

Represents the user accessing the webpage.

### **6.4.1.2 Server**

This is the web server our web app is hosted on. The “hub” of the application where all computation takes place. Is the link connecting the client to the database and is what allows users to access and run the application.

### **6.4.1.3 Language Model**

BERT language model pretrained on over 1 billion training pairs and fine tuned over additional custom data created by us. (Reimers, Nils). Uses a mean pooling method to encode given strings into a data type understandable by statistical comparison methods. In our case, cosine similarity.

### **6.4.1.4 NoSQL DB Mongo**

Database holding the resources. NoSQL so documents of almost any format can be added without issue. Is accessed twice in our program. First is upon initialization of the server. The server will request every resource from the database and encode them, storing these new encoded resources in a dictionary to be accessed later. Next, when a user executes a valid query that has one or more matches in the encoded dictionary, the database is accessed again in order to retrieve all of the necessary information about a resource to be displayed to the user.

### 6.4.2 API specification

This application will likely not integrate any external APIs. The one case where this would be different is if a 3rd party API that handles natural language processing becomes available or is made known to the team. If this happens, this section will be updated to reflect this change.

### 6.5 Behavioral Design

This system is quite simple in its behavioral usage. Because this project encapsulates a component that will one day be added to a larger software and healthcare data ecosystem, there is no need for a behavioral design. The only states in the application would be search entry and search results.

## 7. Evaluation

### 7.1 Overall Evaluation Plan

**Definition of success:** Output of system currently meets expectations for all scenarios manually, code has been reviewed and approved.

**Internal Testing Questions:**

- Code will undergo unit testing and integration testing internally
  - Unit testing: Tests individual aspects of the system, without relying on the dependencies of other parts. For example, if we allowed users to create an account, and wanted to test the account creation through unit testing, we could break it down into several unit tests:
    - Ensure when user clicks “create account”, a banner pops up at the top of the screen
    - When user clicks “create account”, the database updates with a new record
    - Etc
  - Integration Testing: Checks that different aspects of the system are working in tandem. This could involve something like running a simulation of a user creating an account, logging in, and checking that they are listed in the “users” section of the website
- The metrics we will use to ensure our system is adequately tested is a code coverage score of 80%.
- Ideally we can set up a CI/CD pipeline through GitHub to automatically run our tests as code is pushed into the repository. This would likely involve using GitHub-hosted virtual machines (as alternatively we would have to set up a server for automated testing ourselves). If we do take this route, the goal would be to have it set up for the first code submission

### **User Acceptance Test Technique:**

- The sponsor will be providing the customers to us. The first test will likely be very open ended and just ask users to provide various observations and criticism of our system as a whole after using it for a few minutes. Following that, the tests will likely be more guided and designed to gauge responses on very specific aspects of the system.

## **7.2 Internal Testing Plan**

- The three components we will have to test are the database, search engine, and NLP
- Testing environment: we will need unit tests and integration tests
- Materials/resources required
  - The coverage python package (Test Coverage - Flask Documentation)
  - Unittest python package
  - Pytest python package (Testing Flask Applications - Flask Documentation)
- We will mainly be doing functional testing, due to the nature of our product (healthcare/emergency services needing correct information). We want to ensure our product works in situations where the user needs help urgently.
  - Unit testing: we will need to test the search engine and database with separate queries. This can either be done separately or together, since we will be using NLP to turn the queries from 'human' search engine queries into database queries. In this stage of testing (internal), it would be best to test these separately at first.
  - Integration testing: We will also need to test all aspects of the website as a whole, the same way a client would experience it. Because of this, we need integration testing that covers every aspect of the website a user will interact with. (Patrick Kennedy)
- Non-functional testing is also important, since we will need to make sure our product still maintains high performance. As mentioned earlier, we want to make sure the product works in situations where the user needs help urgently, and if our searching is slow this can be detrimental.
  - Performance and load testing: Test response times for queries, test concurrent queries
  - We just use python's built in time module for this
- Regression testing: for any code changes or optimizations resulting from functional/non-functional testing, we will need to retest our product to check if it still works properly after changes

There are many phases of testing, starting with unit (or component testing), which should be part of implementation. The next phase is commonly called integration which involves incrementally testing the system as different components are integrated. The final integration test, with all

components of the system integrated, is often referred to as a system test or validation test. System test is similar to validation test, except for the test data being used, with system test typically involving hypothetical data which should mimic real data. Validation testing should use actual user data, with the test being performed by the development team, using the internal testing environment.

Integration should be a consideration throughout the project. You should not wait to integrate until the end of the project, as this will decrease the probability of having a complete working system by the end of the course.

Interactions likely to cause problems:

- The first interaction will be ensuring that the user's query is encoded properly, and that the filters blocking invalid queries hold

Test Writer: Wyatt			
Test case name	NLP test #1	Test ID #	NLP-01
Description	Checks that the NLP model is able to properly encode the user query to compare to the corpus	Type	White box
Step #1	Action #1: User types in "medication for pain in abdomen" into search	Expected result: Query is encoded as per normal	
Step #2	Action #2: User types "Maternal care near me"	Expected result: Query is encoded as per normal	
Step #N	Action #3: User types in "SELECT * DROP TABLES;"	Expected result: Query is rejected for having invalid tokens, is not encoded, and user is prompted to try another query	
Step #N+1	Action #4: User query is blank	Expected result: Query fails to be encoded and user is prompted to type another query	

- Queries that are accepted and have at least one relevant resource have that resource returned to them

Test Writer: Wyatt			
Test case name	Database Search #1	Test ID #	DS-01
Description	Ensures that queries that are close to existing resource descriptions will have matches, and those that do not will not	Type	Black box
Step #1	Action #1: Query is “I’m a pregnant woman and need local healthcare resources”	Expected result: top result is BCS Prenatal Clinic	
Step #2	Action #2: Query is “I’m running out of food”	Expected result: top result is Brazos Church Pantry	
Step #3	Action #3: Query is “I’m having an emergency”	Expected result: top result is University Emergency Medical Services	
Step #4	Action #4: Query is “Is Popeye’s or Cane’s better?”	Expected result: Nothing is returned	

- Queries can have multiple responses or just one

Test Writer: Wyatt			
Test case name	Database Search #2	Test ID #	DS-02
Description	Ensures that queries that are close to existing resource descriptions will have matches, and those that do not will not	Type	Black box
Step #1	Action #1: Query is “I’m a pregnant woman and need local healthcare resources”	Expected result: BCS Prenatal Clinic, Elizabeth House Maternity Home, Women’s Care Plus, Brazos Valley Women’s	



		Center and Health for All are returned	
Step #2	Action #2: Query is “I’m running out of food”	Expected result: Brazos Church Pantry, Brazos Valley Food Bank, Meals on Wheels are returned	
Step #3	Action #3: Query is “I’m having an emergency”	Expected result: University Emergency Medical Services	
Step #4	Action #4: Query is “Is Popeye’s or Cane’s better?”	Expected result: No results are returned	

## 7.3 User Acceptance Test Plan

### 7.3.1 Recruitment of Users

Our usability tests will aim to determine whether the Intelligent Child product fulfills its primary functions as defined elsewhere in the report.

The exact age, level of technology knowledge, and other demographics of our primary target users, designated Regular Users, for the end product are not yet known, but should be communicated to the Intelligent Children team by the Olivia Health team in time for inclusion in recruitment and the final report. For the time being, we do know that our target user is very likely to be women and girls with a low level of technological experience, between the ages of fifteen and twenty, who either are experiencing their first pregnancy or have given birth to their first child within the most recent calendar year.

There is a possibility that we will also have a secondary user demographic, tentatively designated Administrators, consisting of medical professionals who will use the site to enter information about their practices, or other resources that would be useful to first-time parents. As with the primary user demographic, our team has not yet received information about this group of possible end users from the Olivia Health team, but should know whether this group is to be considered in user acceptance testing (and, if so, what demographics to recruit) in time to conduct testing and include this demographic information in the final report.

Across both possible roles, we will endeavor to exclude participants who do not fluently speak English, due to our inability to provide documents in languages other than English. If possible, we hope to exclude participants who are younger than 18 to avoid additional paperwork, but this will depend on the exact nature of the Regular User profile provided by the Olivia Health team.

To recruit users, it is most likely that we will use an email recruitment script. This will be most useful if we can obtain access to the tamu-opt-students mailing list, to reach a large number of people in the target age range for Regular Users. We may also be able to reach out to the Aggie Pregnant & Parenting Student Organisation, an officially recognized student organisation at TAMU, for more targeted recruitment of individuals who have a higher chance of meeting the criteria to participate as a Regular User. To express interest in participating, potential participants will fill out a Google Form, the link to which will be included in the email.

## **7.3.2 User Acceptance Test Artifacts**

### **7.3.2.1 Protocol**

#### **Overview of UAT:**

- **# of participants:** 20
- **Components they interact with:** Only directly interact with the search bar and maybe some filters
- **Pre-survey:**
  - Give the customer an outline of the goal of the website; describe use cases.
  - Give customer some initial queries to demonstrate functionality, then come up with their own queries they would find useful personally
- **Observation Protocol:**
  - Does using the app as it should seem to frustrate the user?
  - How long does it take to execute a query?
  - Do they seem pleased by the results of their queries?
  - Do the queries they come up with have existing answers?
- **Post-survey:**
  - Simply compile the observational data and survey results
    - The observational data will not be tied to any specific user, just a collection of observations from anonymous users
    - Survey data can also be anonymous, holding data from specific users would not be necessary

### **7.3.2.2 Survey**

In order to get feedback after each participant performs their UAT, we will use Google Forms to collect responses to their questions. In this survey, there will be two distinct sections: quantitative analysis and qualitative analysis. This survey will be 15 questions long.

Quantitative:

- The first ten questions of the survey will be asking participants to rate certain aspects of the application on a scale of 1-10, with 1 representing a failure and 10 representing a success. Using these scores, we can track compliance with the specific goals of the project. Some example questions are listed below:
  - On a scale of 1-10, how intuitive was the application to use?
  - On a scale of 1-10, how relevant were the presented resources to your search?
  - On a scale of 1-10, how confident are you in the accuracy and validity of the resources provided?

#### Qualitative:

- The final five questions of the survey will be open ended/free response to allow each user to explain some of their thoughts while using this application. While this section will not be useful quantitative analysis, this will hopefully provide insight into the trends in the earlier quantitative section. Some example questions are listed below:
  - Was the website easy to navigate and use? Please explain if any part of using this application was difficult or confusing.
  - Were the search results relevant to your question? If not, please give an example of a resource that was not relevant to the search you entered.
  - Is there any other feedback you would like to give the team? Please enter any other opinions you have that were not asked in another question on this survey.

Due to the nature of the agile development process, we will likely not have every functionality implemented in the first UAT round. In this case, the survey will be cut down and the questions will be refined to match the MVP submitted for testing.

#### 7.3.2.3 Interview

The interviews for UAT will be minimal as the surveys will make up the bulk of feedback collected. However, we would still like to speak with the users who used the software as not all feedback can be encompassed within a survey. There will not be any specific questions for the interview section; rather, the team member administering the UAT will ask questions pertaining to the user's specific experience. This will be done *before* the survey is completed.

Example: A user opens the application and can't figure out how to run the search (search button isn't easily noticeable, confusing UI...). After the test, the administering member will ask about the specific difficulties around finding the search button.

# 8. Implementation

## Overview

A user loads the index page, types in a query, and hits search. A list of (up to 5) resources are returned to the user with various information about the resources, such as the name, description, phone number, and street address (if applicable). Pictured below is the first thing a user sees accessing the website, then below that shows the output a user might expect to their query.

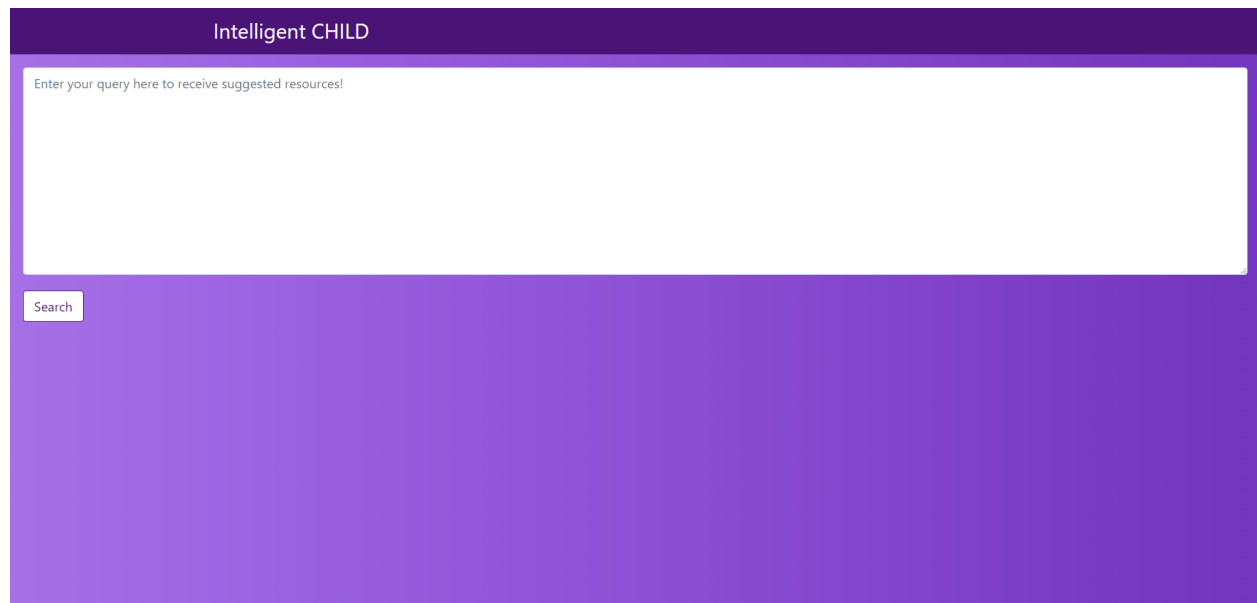
The screenshot shows the index page of a website titled "Intelligent CHILD". The page has a dark purple header with the title in white. Below the header is a large white search input field with the placeholder text "Enter your query here to receive suggested resources!". To the left of the input field, there is a small white button with the text "Search". The rest of the page is a solid dark purple color.

Figure 1: Screenshot of index page of website

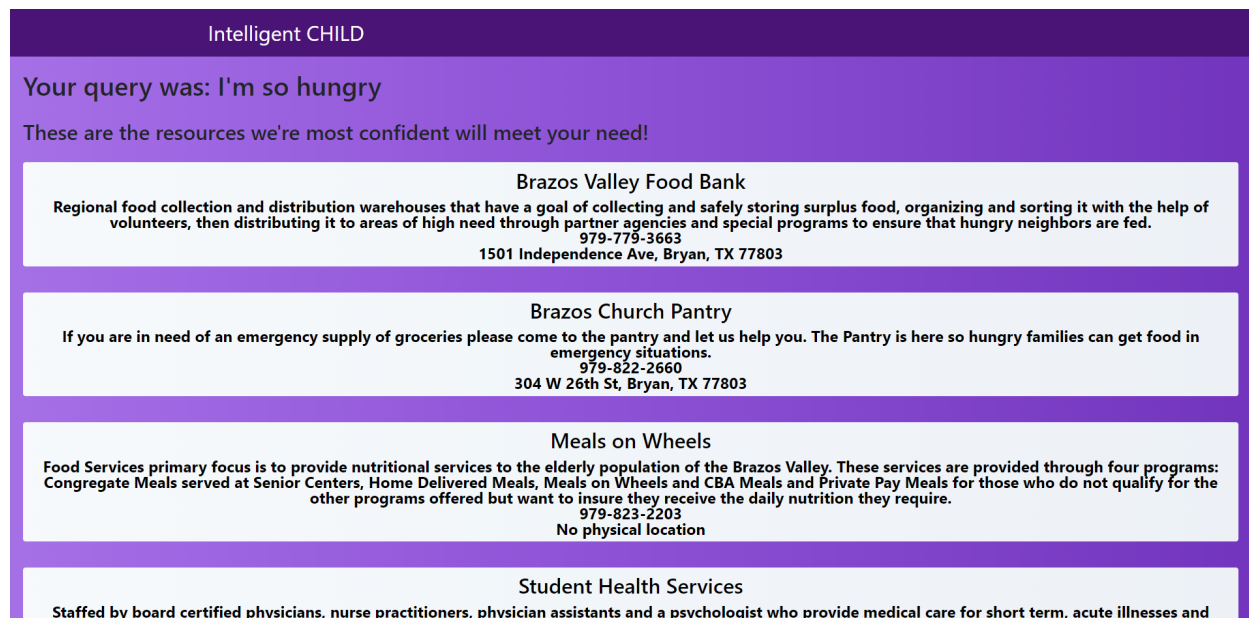


Figure 2: Response to user query

## Database

To store these resources, we use MongoDB which is a NoSQL database. A NoSQL database stores resources in the form of “documents”, where each document is an object containing several fields that can be unique to itself or shared with other records. This means in a database of resources like ours, if a new resource to be added is missing a field, or has some unique field to be handled in the backend code, it can be stored in the same database as our other records with no issues. Additionally, each document is independent of all other documents in the database, there are no relations connecting one document from one table to another document in another table.

Over the course of this project we were given raw data in the form of a csv file. We created a script to preprocess the contents of this csv and store the results in our Atlas MongoDB database. An Atlas MongoDB database is simply the cloud version of a MongoDB database. The preprocessing simply consists of creating a “Preprocessed Description”: Combining the already-existing description field of a record with the name of the organization, a title given to the resource, and the city the resource covers. This was accomplished using a pandas dataframe to read the csv and create the new column. The dataframe was then converted to a dictionary and inserted into the Atlas MongoDB database using the pymongo library. We noticed a slight improvement of the results with this preprocessing.

## Model and Training

Once we have all of these resources nicely tucked away in a MongoDB database, we need a way for the users to actually search through them. We decided to use the sentence-transformers python package to accomplish this. Below are two figures demonstrating an overview of this functionality

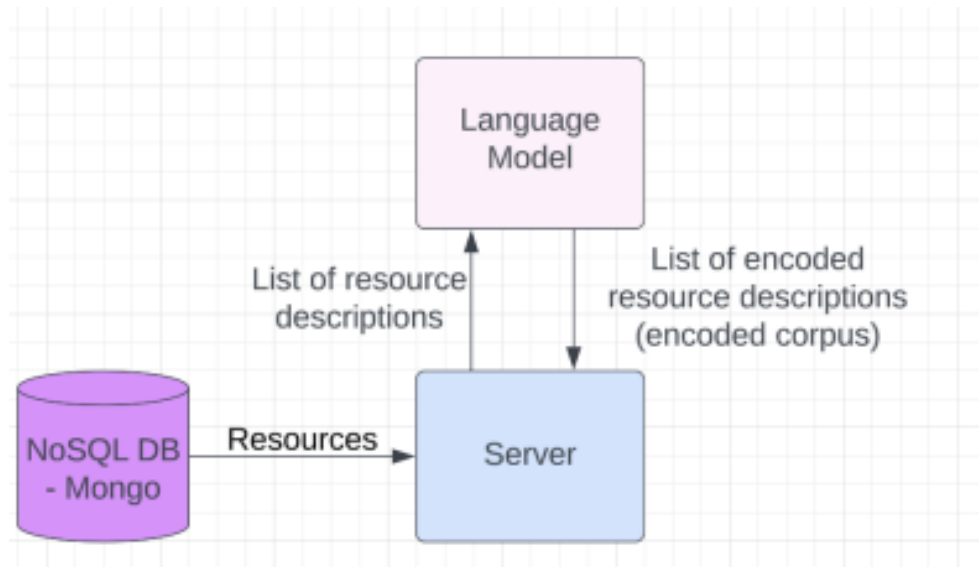


Figure 3: Encoding the corpus

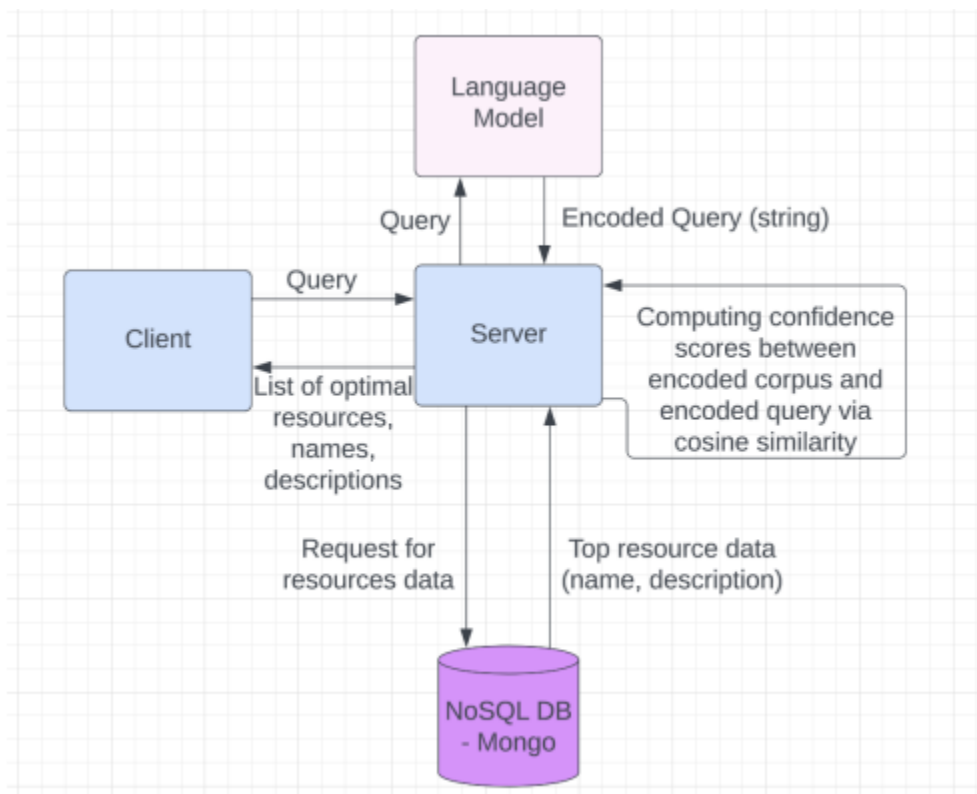


Figure 4: Encoded query and corpus comparison

The figure 3 above demonstrates how we encode the corpus.

- Upon initialization of our program (when the files are uploaded to the hosting service and it is executed)
- We pull all records from the database into a list
- Then encode them using the language model.
- We then save this list of encoded records on the server itself and any time a user makes a request to the website, the request has access to the encoded corpus (it is shared between users).

Figure 4 demonstrates the actual search component of our website.

- User inputs query
- Query is fed to language model and encoded
- Server loops through encoded corpus and does cosine similarity between encoded corpus and query, returning the top k (maximum of 5)
- The database is re-accessed using the preprocessed description of the top 5 resources to get even more information to display to user
- Top 5 resources over a specified confidence threshold (0.3) with the new information are displayed to the user

The language model used is the Sbert model, a python NLP implementation of the BERT language model from Google. We specifically used the all-MiniLM-L6-v2 version of the model as our base, which was pre-trained on more than 1 billion training pairs and is 5 times faster than some of the general-purpose language models. We ran into some issues where this model was too general for our specific use case so we came up with an idea to fine-tune the model using our own training data.

Creating the training data was a lengthy process that involved manually coming up with queries (using the data we had from UAT 1), finding a matching description in the database (or not matching at all) and giving a score from 0 to 1. We managed to create a total of 250 training examples for the model in csv format, with the format being [query, description, score]. We leveraged sentence-transformers built-in fit() method to fine-tune the model with our custom data and save it in a directory to be accessed easily on the website. Below is a figure showing a small sample of our training data.

1	I'm struggling with drinking too much and can't quit.	A drug and addiction treatment center to help fight addicton.	0.9
2	I need help quitting drugs.	A drug and addiction treatment center to help fight addicton.	0.9
3	Can you drink during pregnancy?	A drug and addiction treatment center to help fight addicton.	0.5
4	Where can I find an ultrasound?	A drug and addiction treatment center to help fight addicton.	0.2
5	Watermelons are on the side of the street after dusk.	A drug and addiction treatment center to help fight addicton.	0
6	Can prescription drugs be addictive?	A drug and addiction treatment center to help fight addicton.	0.7
7	I've been drunk every day for a month and need help.	A drug and addiction treatment center to help fight addicton.	0.9
8	Where can I find prenatal care?	Cover a multitude of practice areas including consumer, landk	0.2
9	I can't afford groceries right now.	Cover a multitude of practice areas including consumer, landk	0.2
10	I can't afford legal costs for a speeding ticket.	Cover a multitude of practice areas including consumer, landk	0.8
11	How can I sue someone for sexually assaulting me?	Cover a multitude of practice areas including consumer, landk	0.6

Figure 5: Sample of training data

## User Interface/ Hosting

List of required packages for the virtual environment:

- Flask (web framework), version 2.2.3
- Pymongo (MongoDB integration), version 4.3.3
- Sentence\_transformers (NLP engine), version 2.2.2
- PyTorch, version 1.13.1
- NumPy, version 1.24.2
- Pytest, version 7.3.1
- Coverage, version 7.2.3

We decided to use Flask as both our web framework and user interface package. This allowed us to control the various routes and endpoints of the website while keeping everything encapsulated in a single main `db_search.py` file. It's as simple as defining a route, defining a function for that route, and a set of processes that take place in that function before the template is displayed for that route. For example, in our main `db_search.py`, we have 2 routes defined:

- An empty route (`/`) that users hit when simply accessing the website.
- Simply renders the `index.html` template and does nothing else.
- A results route (`/results`)
- The function defined for the route does everything described in figure 4 takes place before `results.html` is rendered.
- Anything implemented in `db_search.py` outside of a route is run upon initialization and never again (in our case encoding the corpus).

We decided to host on a website called `pythonanywhere.com` for both of our user acceptance tests. Setting it up was as simple as uploading our main `db_search.py` file, the fine-tuned model, and the templates to be rendered to our site directory. We then had to set up the virtual environment by installing the necessary packages listed above. Additionally, we had to make changes to a WSGI file to allow the flask app to recognize this new hosted version. We did run into some issues in UAT 2 with our app timing out every 24 hours or so, and determined that it had to do with our model wanting to create multiple threads to encode the query, and the `pythonanywhere` service doesn't allow thread creation on web apps. This shouldn't be a problem for other hosting solutions, however.



# 9. Results

## 9. Results

### Baseline

The Intelligent Child project is a novel, curated searching experience that combines several existing technologies to provide a more specialized, efficient, and reliable search engine for community resources and healthcare providers. The project is split into two main components: the data design, which involves creating a curated database of resources, and the semantic search engine, which allows users to search through the database using natural language inputs. To evaluate the performance and reliability of the Intelligent Child system, it is essential to establish appropriate baselines for comparison.

One baseline that is particularly relevant is a search engine like Google, which also uses semantic searches via natural language inputs. However, the Intelligent Child system differs from Google in that it focuses on a much more specialized set of data, including community resources and healthcare providers. This means that the system needs to be able to finely search over a smaller set of data using similar semantic inputs. By comparing the Intelligent Child system to Google, we can evaluate the system's ability to provide more targeted and accurate results for its users. In addition to search engines like Google, other baselines can also be used to evaluate the Intelligent Child system's performance and usability. For example, searches performed over smaller sets of curated data typically do not involve NLP or confidence-ranked searches, but instead involve straightforward parameterized searches that are performed by someone with a level of knowledge or skill with the system. These baselines are particularly relevant for evaluating the Intelligent Child system's usability and accessibility, as they represent a more familiar and accessible form of searching for many users.

Overall, by establishing appropriate baselines for comparison, the Intelligent Child project can evaluate its performance, reliability, usability, and accessibility against existing technologies and approaches. This will help ensure that the system meets the needs of its users and provides an efficient and effective way to access vital community resources and healthcare providers.

### Conditions

The Intelligent Child project is a cutting-edge search engine that leverages curated databases to provide users with a more reliable and user-friendly search experience. Unlike other search engines that may pull data from a variety of sources, our solution relies on pre-vetted databases to ensure that users are presented with accurate and trustworthy information. By doing so, we aim to provide users with a more trustworthy source of information that is not influenced by deceptive search engine optimization strategies.

In addition to being more reliable, our solution is also more user-friendly, particularly for users with low technical expertise. We believe that usability is a key factor in the success of any search engine, and we have designed our solution with this in mind. Our aim is to reduce the time it takes for users to familiarize themselves with the system and search for real-world information that is relevant to their lives. By doing so, we aim to provide a more accessible and inclusive search experience for all users.

Furthermore, our solution is designed to be easy to maintain and manage for the team who will assume ownership of the project. Our flexible NoSQL database is specifically designed to support different data types, regardless of their similarity to existing data types. This means that system administrators can easily host several different data types within the same database, making it a more convenient and flexible solution to manage.

In conclusion, the Intelligent Child project is a revolutionary search engine that prioritizes reliability, usability, and manageability. By leveraging curated databases and a user-friendly interface, we aim to provide users with a more trustworthy and accessible search experience. We believe that our solution represents a significant improvement over current search engine offerings and are excited to see it in action.

## **Metrics**

When evaluating the effectiveness of the Intelligent Child search engine, there are several key factors to consider. One important factor is the time taken for results to be returned after a query. Our system aims to provide fast and efficient results to users, without sacrificing the accuracy and relevance of the results.

Another important factor is the relevance of the results when compared to the query. Our system leverages advanced search algorithms and natural language processing techniques to ensure that the results are highly relevant to the user's query. By doing so, we aim to provide users with the most accurate and helpful information possible.

The actionability of the results is also a key consideration when comparing our system to baselines. We believe that our system provides highly actionable results that can help users find the information they need quickly and easily. Our goal is to ensure that users are presented with information that is not only relevant but also actionable, allowing them to take action based on the results of their search.

Finally, we believe that the top result returned by our system is typically the most relevant to the user's query. Our advanced search algorithms are designed to prioritize the most relevant results, ensuring that users are presented with the information they need as quickly as possible.

Overall, our system aims to provide users with fast, relevant, and actionable results that are highly accurate and reliable. By leveraging curated databases and advanced search

algorithms, we believe that our system represents a significant improvement over current search engine offerings and are confident in its ability to meet the needs of users in the community.

- **Evaluation Results:** Results often can be categorized as follows:
  - **Quantitative Results:** Results / outcomes are often presented numerically and presented with tables or graphs. Examples include speed and accuracy.
  - **Qualitative Results:** These results are often presented with scaled responses and supplemented with excerpts of responses. Examples of qualitative results include user preference of specific conditions.

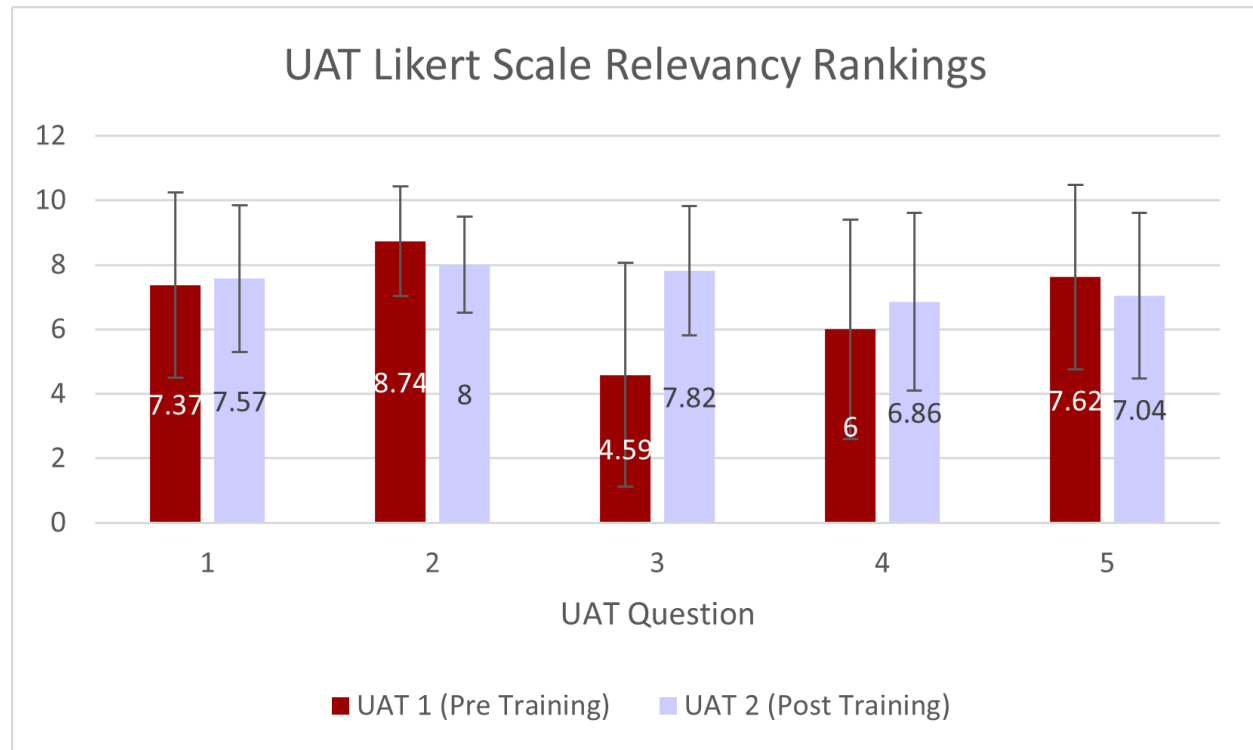
## Results

For UAT 1 and UAT 2, the questions asked by each user were the same in some cases. For these questions, this section will compare the results between each UAT. In two distinct sections, the users had to a) input a specific query, and b) input a query based on a given hypothetical situation. The average rating is given in the table below based on a Lickert scale rating to determine the relevancy of the returned resources. As instructed, a “1” represented the application returning resources that were not at all relevant, while a “10” represented the application returning relevant resources. The specific queries and scenarios are also listed below.

1. Where can I find prenatal vitamins?
2. I’m pregnant and having really intense back pain. Where should I go?
3. (Hypothetical Scenario) You are having difficulties paying for your meals. You realize that you won't be paid for another two weeks but don't have enough money in your account to go grocery shopping.
4. (Hypothetical Scenario) You are 18 weeks pregnant and have begun to feel incredibly anxious about childbirth as the due date fast approaches. Just last week you had your first ever panic attack and now are looking for a way to manage your anxiety.
5. (Hypothetical Scenario) As a new mother, you aren't sure about how best to feed your newborn. You've been looking online for information about formula and breastfeeding, but would rather find a local resource in the BCS area to help guide you.

UAT Question	UAT 1 Relevancy Score Avg	UAT 2 Relevancy Score Avg	UAT 1 Std Deviation	UAT 2 Std Deviation
1	7.37	7.57	2.88	2.28
2	8.74	8.00	1.70	1.49
3	4.59	7.82	3.47	2.00
4	6.00	6.86	3.41	2.76

5	7.62	7.04	2.86	2.56
---	------	------	------	------



Some of the discrepancies in lower scores between UAT 1 and UAT 2 can be described by the difference in results returned during the two tests. While UAT 1 only returned one result (meaning we knew the predetermined search's results before sending the test out), UAT 2 returned up to 5 relevant resources to a user's query. Although the UAT 2 responses were much more likely to contain a relevant resource, there was still a chance of one or two less ideal resources being presented which may lower the score. Notably, the distribution of Likert scores during UAT 2 was much more precise - UAT 1 had many 1s and 10s while UAT2 did not.

We also asked users about our search engine as a whole. Below are the results in terms of the Likert scale:

UAT Question	UAT 1	UAT 2
Were the resources returned to you overall relevant to your searches?	6.11	6.96
Did you feel that you could trust the provided resources?	6.89	8.07
Would you use a platform like the IChild system over Google	4.52	6.18

to search for medical or community resources?		
---	--	--

For the anonymized raw UAT data, please click the links below.

UAT 1 Results:

[https://docs.google.com/spreadsheets/d/1BXUDFb6t3P-uZFEe6zSETI6\\_2M1bER5hCWwApLvH37g/edit#gid=216984194](https://docs.google.com/spreadsheets/d/1BXUDFb6t3P-uZFEe6zSETI6_2M1bER5hCWwApLvH37g/edit#gid=216984194)

UAT 2 Results:

<https://docs.google.com/spreadsheets/d/1eacyg55pATzsBjbdjnvCrm3D27Y0o2DfX5ML3-dstSk/edit#gid=1195950997>

## Definition of Success

Below we have included a few representative user stories along with their definitions of success, followed by a brief explanation as to how these definitions of success were accomplished within the project.

User Stories:

1. As an expectant mother, I should be able to search for resources in my own language because I am not familiar with medical terminology.
2. As an expectant mother, I should only receive relevant resources from my search because I do not have time to parse out unwanted results.
3. As an expectant mother, I should only see a handful of curated resources because too many may overwhelm or confuse me.
4. As a young expectant mother, I should be able to understand how the application works easily.
5. As an expectant mother, I should be able to search for financial aid resources.
6. As a young and expectant mother, the searches need to be executed quickly and not take very long.

Definitions of Success:

1. Users should be able to put a query in with natural language and be returned a resource if a relevant one exists
2. Queries with no connection to any resource will not have any resources returned.
3. Queries return 0-5 results based on the relevancy of resources in the database.
4. The application is deemed “easy to use” by at least 80% of users rating the application at least an 8 out of 10 on a Likert scale questioning if the website was intuitive and easily understandable.
5. During UAT, user questions about the relevancy of resources returned when asking about financial aid should have at least a 75% positive response rate.
6. Queries should ideally execute in under 1 second.

Explanation for meeting the definitions of success:

1. Using the Sbert language model and additional training to tailor the model towards medical and community resource searches, users can access resources via searches.
2. When no resources meet the confidence score threshold of 0.3, the website will display that no relevant results have been returned.
3. Using the confidence score threshold of 0.3 up to 5 resources will be returned and none will be returned if no resources surpass the threshold.
4. During UAT2, 24/28 users (85.7%) gave the application at least an 8 out of 10 on a likert scale asking about ease of use. 16 users (57.1%) rated the site as a 10 out of 10 for ease of use.
5. During UAT Round 1, 21/27 users (77.7%) deemed the responses from a prompted question referencing a need for financial aid as having a positive relevancy score.
6. As determined by internal unit testing, the average query time for a search to execute is 0.3 seconds.

## 10. Discussion

### Insights

Throughout the development of the Intelligent Child system, we evaluated and implemented under an agile framework. This allowed the team to ensure compliance with the customer requirements while also requiring us to be very reactive - one such instance comes from the very beginning of the project. While originally planning on using a SQL relational database for storing information, we learned from our technical mentor that a NoSQL database would be more applicable for NLP and more scalable for future work. The team needed to redo approximately two weeks of design in a very short timeframe - working under the agile framework allowed us to quickly implement and refine these new data designs. Overall, the agile development framework was a powerful tool in completing this project.

As for final insights as to the efficacy and success of the Intelligent Child system, we look towards UAT results. A significant portion of users said they would prefer a system like ours over a classical search engine (Google) while the prototype was still relatively barebones - this speaks volumes to the potential in this application once refined and searches become even more accurate and personalized. Throughout the training of the model, search results did become significantly more relevant and helpful; however, there are still certain cases which testing has shown do not work properly. For example, a person asking the engine “where can I buy a lakehouse” should not yield any results, but the engine returns resources related to affordable housing. This may be due to low amounts of data in the database or a model that has not been

sufficiently trained. Another possible answer is the need for a response rejection algorithm within the system.

One particular area where the team could have improved on was communication and scope setting with the customer. This system was implemented almost like a “start up”, with quickly changing criteria and requests from the client. Having a more concrete understanding of the tools available to the team would have prevented a lot of headache - two main examples of this being hosting and data availability. We had originally planned to host natively on the customer’s website, however we were unable to due to strict permission restrictions within their own server environment. As for data availability, we were not provided a sufficient dataset of local resources and had to fill out the corpus ourselves (possibly injecting unknown biases).

## **Team Interactions**

Overall, the development team worked very well together. We had a few people with assigned roles - for instance, Emmeline was in charge of communicating with the customer and Wyatt was in charge of “quality”. Mainly, we all worked together on everything to ensure that each member had a thorough understanding of the requirements, progress, implementation, and design. We had frequent pair coding sessions during implementation and designed the system in meetings with the full team - very little work on the project was done individually with the exception being documentation. When completing the necessary project documentation, team members would split different sections to work on then review the other member’s work once finished.

If we were to start the project from scratch, having more significantly defined roles would have been helpful. The caveat to this would be that the team would not know one another’s strengths and weaknesses at the very beginning of the project. If working within a new team, not much would likely change between this implementation and a new one; however, a new project completed within the same team would likely see more defined roles and responsibilities.

## **Safety Concerns**

Due to the Intelligent Child system dealing with real maternal and familial health outcomes, there are significant risks to safety when deploying this software. During testing and development, there were no *actual* users - i.e. people seeking resources and referencing our site to do so. The accuracy and relevancy of searches are incredibly important as we do not want to persuade future users into utilizing resources not relevant to them, especially when better alternatives may exist. This could have potential negative health outcomes as accessing more helpful resources may be delayed due to misinformation.

In order to help the safety of this application, the team trained the NLP model over relevant queries and resources harvested from our user acceptance testing. While this training

drastically improved the accuracy of responses, there were still edge cases or confusing queries which could yield a less than optimal result. For future work, additional training is a must.

Another important aspect in the safety of this system is ensuring all resources and information in the database is helpful, trustworthy, and vetted by the Intelligent Child health team. When the team needed to fill in the database with additional resources, some were omitted due to their negative health outcomes (for example, pregnancy crisis centers which do not provide healthcare but are often associated with healthcare).

Another important note for this project is an issue that may be encountered by many programmers or people within the tech world. There's an old adage that says "when you're holding a hammer, everything looks like a nail." Similarly, when you work in technology, the potential for technology solutions appear everywhere. When it comes to real positive outcomes for maternal and familial health, there are much better steps to be taken than creating a resource search database - mainly changes in policy and healthcare costs. While this does not directly represent a safety concern, it may speak to a misdirection of where to look for improvements to the health and safety of potential users.

## **Testing**

The Intelligent Child system was thoroughly tested to ensure compliance and accuracy. We included unit testing to prove that the system functioned correctly, but the main testing was with users. Through a two round UAT process, we were able to identify weaknesses in our implementation and fix them before handing the project off to the customer. The system does currently work within the bounds of the scope and criteria provided by the customer; however, there are still cases in which testing revealed shortfalls. The main area needing improvement is the accuracy of the search - although ~95% of queries worked as anticipated, strange wording or edge-cases often resulted in invalid responses still being displayed to the user.

Should this project be repeated, additional UAT is the most important inclusion. Because of the highly personalized nature of this application (each user queries however they feel most comfortable), gathering large amounts of data on possible query strings or parameters is paramount.

## **Large Scale Implementation**

There are three main areas needing to be addressed for large scale implementation. First, additional data is a must. In order to parse this data depending on physical space, location tools such as the Google Maps API could be utilized. This would also necessitate additional resources from various regions wherever this is to be deployed.

Second, additional training for the model is very important. Although training has yielded impressive improvements in search queries, additional training is required as the system is not



yet robust enough to handle edge cases or strange queries without being confident enough to decide not to display potentially erroneous results.

Finally, scaling this project would require an “admin portal” to add data into the system. As the system stands, data is added via a python script connected to the Atlas DB instance - having a GUI would make this process feasible for non-power users.

## **Existing Solutions**

No existing solutions closely match the Intelligent Child implementation. Furthermore, the technologies utilized were all open source. There is no threat to patents posed by this implementation.

## **Model Training Shortfalls**

Our NLP model failed in a few specific circumstances that must be addressed by future work if this system is to continue and be deployed to real users.

1. Nonsensical Queries. Although mostly filtered after training, a user can occasionally input an erroneous or nonsensical question which will still yield a response. Interestingly, the most commonly returned resources for “random text” were related to substance abuse. Some specific and targeted training could fix this specific case, but a form of “denial” algorithm may be necessary.
2. Edge cases. Some queries are not represented by resources in the database but are asked in a way similar enough to prompt a response. One instance is a user querying “Where can I buy a lakehouse” and being returned resources for affordable housing. This could potentially be fixed, again, by a form of “denial” algorithm to prevent erroneous results from displaying. Of course, additional training would be very helpful.
3. Small data set. Because our provided data was so minimal, the training was hyperfocused on these specific resources. Additional resources added into the database must compete with existing biases towards present resources, so retraining is essential.

# **11. Future Work**

In terms of future developments for the Intelligent Child Application, there are many opportunities for significant improvements. This can be broken down into a couple of different categories.

First, additional model training is paramount to the improvement of the search and results experienced by users. Because this application has the potential to affect legitimate changes

(positive or negative) to maternal and health outcomes wherever it may be deployed, an incredibly accurate search is necessary. When additional data is inserted into the database, retraining must occur as training completed to this point will have a bias towards existing resources. In each training triplet, the second value, or the “result”, was represented by the description of each resource matching a query after being sanitized of names and locations. Perhaps a system to generate training data at a higher frequency would be very helpful. An integration of active learning by the deployed application (training by real user interaction and feedback) could also be implemented to further automate this process.

On the topic of search accuracy, one additional feature that will be required to implement for further deployment is some form of location/ geographical closeness feature. Although the database is currently localized to the Bryan/ College Station area, users may be accessing these resources from vastly different locations. A user in Dallas should not be presented with a clinic in Austin to visit, nor vice versa. Some suggestions would be to research the Google Maps API, implement the location data from a user’s browser or machine, and add additional database filters such as coordinate points for distance calculations.

Second, the Intelligent Child Application is to be eventually implanted within a larger piece of technology for Olivia Health - the “Ollie” chatbot. The system currently does not have contextual understanding (cannot ask multiple questions), so a chatbot would need significant engineering to interface with the created semantic search. The Ollie chatbot would also need to be able to reference informational and educational materials posted on the Olivia Health website, which would not require location data or contact information. The noSQL database design allows for flexibility in types of data that may be presented to the user, but refactoring of the database would be required for Ollie to reference these other materials when necessary. One particularly interesting problem to solve is for the system to determine whether a user needs information that can be provided by these modules or if they need information pertaining to a local resource. For example, a new mother searching the site for information about breastfeeding would likely not need a local resource but instead an educational one; on the other hand, a mother asking for help with intense pain would be a more difficult decision. Would an informational article suffice, or would they need to be directed to a local physician’s office?

Finally, there must be additional research into the effectiveness of a search engine like this on real health outcomes. Significant resources have already been allocated to this project without prior research into potential benefits; from a boat-benefit analysis standpoint, research into possible efficacy is necessary to justify continued development. When the expressed goals are to improve maternal and familial health outcomes, there is already significant and fairly uncontested research stating that reduced healthcare costs, greater access to transportation, and lessened restrictions on specific procedures (abortions, hysterectomies, birth control, ...) are what will provide the greatest benefit.

## 12. Conclusion

In conclusion, we have developed a curated search engine that provides accurate and reliable resources on maternal healthcare and pregnancy-related topics. Our advanced algorithms gather and classify relevant information from dependable sources, ensuring the availability of high-quality resources. With our natural language processing capabilities, users can enjoy a personalized and user-friendly experience. Through our two-fold validation approach, we were able to optimize the search engine's performance and improve its functionality. The feedback received from users during the user acceptance testing helped us refine the search engine further, ensuring that it meets the needs of its diverse audience. With our search engine, users can navigate the intricate realm of information with ease, confident that they are accessing valuable insights and guidance. Overall, our innovative approach prioritizes quality, accuracy, and user-friendliness - offering a reliable and efficient solution to the challenge of finding trustworthy resources on maternal healthcare and pregnancy-related topics.

## 13. References

List here the collective references from each member, which should be at least 5 references per team member. Acknowledge all other documents that you used as references throughout the text. Please follow the regular APA citation standard for references.

*Sentence Embeddings and Transformers*. (n.d.). Pinecone.

<https://www.pinecone.io/learn/sentence-embeddings/>

The Google Gospel of Speed. (n.d.). Think with Google.

<https://www.thinkwithgoogle.com/future-of-marketing/digital-transformation/the-google-gospel-of-speed-urs-hoelzle/>

MOZ. (2019). *Page Speed*. Moz. <https://moz.com/learn/seo/page-speed>

*Natural Language Processing (NLP) - A Complete Guide.* (n.d.).

Www.deeplearning.ai.

<https://www.deeplearning.ai/resources/natural-language-processing/>

Rajput, A. (2020, September 3). *Semantic Search Engine using NLP*. Medium.

<https://medium.com/analytics-vidhya/semantic-search-engine-using-nlp-cec19e8cfa7e>

Kennedy, Patrick. (2021, December 14). *Testing Flask Applications with Pytest*.

TestDrivenIO.

[TestDriven.io. https://testdriven.io/blog/flask-pytest/](https://testdriven.io/blog/flask-pytest/)

*Testing Flask Applications - Flask.* (n.d.). Flask.

<https://flask.palletsprojects.com/en/2.2.x/testing/>

*Test Coverage - Flask Documentation.* (n.d.). Flask.

<https://flask.palletsprojects.com/en/2.2.x/tutorial/tests/>

*Databases and Collections.* (n.d.). MongoDB.

<https://www.mongodb.com/docs/manual/core/databases-and-collections/>

Reimers, Nils. (2020) *Pretrained Models*. Sbert.

[https://www.sbert.net/docs/pretrained\\_models.html](https://www.sbert.net/docs/pretrained_models.html)

*Python Unit Testing Framework*

<https://docs.python.org/3/library/unittest.html>

*Training SentenceTransformers Model*

<https://huggingface.co/blog/how-to-train-sentence-transformers>

*Data Augmentation in NLP*

[https://maelfabien.github.io/machinelearning/NLP\\_8/#](https://maelfabien.github.io/machinelearning/NLP_8/#)

*SBERT Loss Functions*

[https://www.sbert.net/docs/package\\_reference/losses.html](https://www.sbert.net/docs/package_reference/losses.html)

*Query Generation for Limited Data*

[https://www.sbert.net/examples/unsupervised\\_learning/query\\_generation/README.ht](https://www.sbert.net/examples/unsupervised_learning/query_generation/README.html)  
[ml](#)

# Annex 1: Project Plan

In line with our sponsor's preferences, we intend to use an Agile development process.

## A1.1 Implementation Schedule

### Overall Scope:

This project consists of three main components. First is the design of the database which will need to be quite robust in order to support intelligent search. Second is the implementation of an intelligent search feature to parse through the database given specific tags or parameters. The third and final component would be a method of natural language processing in order to convert search queries from users into data for parsing the database.

### Database Design:

The database design will be contingent upon the exact fields and methods of search that the customers at IntelligentCHILD require. Since this database will include various community and support resources ranging from healthcare providers to food banks, the database will need to have ample information and flexible columns to accommodate the multiple different use cases.

Columns: One of the project members, Pranav, emphasized the importance of creating intuitive and sensible columns for the database. This will require two main phases:

- Gathering requirements from the customer about the specific columns
- Designing a data layout given these columns
- Implementation of the specific columns in a SQL database, likely to be hosted on AWS

### Implementation:

- Gather requirements such as framework and tech stack, data typed being used, and acquire a data sample
- Go through design iterations, checking with Nick and Professor Wade to ensure the data is laid out in a scalable manner
- Implement the database, either through a DBMS such as PostgreSQL or another framework if required by the customer

### Intelligent Search:

The database must be easily and efficiently searchable; therefore, a system to construct and run queries must be designed. Since the searches will be generated via natural language processing, the process by which a search is performed needs to be robust, thoroughly tested, and flexible.

#### Design/Planning:

- Using the data design, construct a process by which searches can be performed easily (whether this is through tags on the data, complex SQL queries, or some other method is yet to be decided)
- Gather information from the customers in order to determine the search heuristics

#### Implementation:

- Connect the search program to the SQL database that is likely to be hosted on AWS
- Implement the actual search algorithm parameters - the exact parameters of the search (tags, strings, ...) are not yet decided.
- Implement the search algorithm heuristics - the exact heuristics are not yet decided.

#### Testing:

- Thoroughly test each iteration of the search algorithm implementation over a variety of data sets and search queries
- Using customer feedback, compare the results of the data search tests to the expected results. For example, if a search is supposed to return a list of healthcare providers specializing in maternity care within the Brazos Valley, ensure that the results a) contain all of the expected resources, b) are displayed in the correct order, and c) have no extraneous outputs.

### Natural Language Processing:

The end user for this project is likely to be a person typing in a search query into an app or website on their phone. The goal is for a query, such as “I need help finding a gynecologist that accepts medicaid”, to return a list of usable and reliable resources. As such, the natural language entered by the end user must be parsed for information necessary to perform a search over the database. We also anticipate this to be the most difficult portion of the project, meaning that this work breakdown could change at any time.

#### Design/Planning:

- Researching various methods of natural language processing, likely with assistance from our TA Nick.

- Looking into AI licenses that can produce tags from natural language inputs
- Designing the structure of the natural language processing program (possibly integrating a 3rd party AI that already has the language processing functionality)

#### Implementation:

- Connect our system to a 3rd party AI
- Connect the natural language processing system to our database search program

#### Testing:

- Using customer feedback, compare the results of the natural language database searches to expected search results
- Ensure that the language processing produces the correct and necessary tags for searching through the database
- Test for various different “types” of natural language - i.e. misspelled words, slang, references to a specific idea or item without being explicitly stated

\* Again, it should be noted that the natural language processing is the most daunting and difficult portion of this project. This plan is currently very tentative and is likely to change and become much more refined with further work and research.

#### Sprints:

We plan on using an agile development methodology so our work will be broken into sprints. The sprints will last around 2 weeks each, and at the end of every sprint, we will turn in a minimum viable product to the client for feedback. These minimum viable products will vary depending on what we accomplish during the sprint, but will be incrementally improved as we continue to make progress.

#### Schedule:

- Sprint 1 (2/6 - 2/16) - Database Design, Search Design
- Sprint 2 (2/20 - 3/2) - Database Implementation, Search Implementation, Natural Language Processing Design
- Sprint 3 (3/6 - 3/16) - Database Testing, Search Implementation and Testing, Natural Language Processing Implementation
- Sprint 4 (3/20 - 3/30) - Search Implementation and Testing, Natural Language Implementation and Testing
- Sprint 5 (4/3 - 4/13) - UAT Changes, Report
- Sprint 6 (4/17 - 4/27) - Preparations for Capstone Expo, Final Report



## A1.2 Division of Labor and Responsibilities

Deliverables (due date):

1. Report
  - a. Project plan (Feb 3rd)
  - b. Gantt chart (Feb 3rd)
  - c. References (Feb 6th)
  - d. Related work
    - i. Part 1 (Feb 6th)
    - ii. Part 2 (Feb 13th)
  - e. Engineering standard (Feb 13th)
  - f. Requirements (Feb 13th)
  - g. High-level design (Feb 20th)
  - h. Evaluation plan (Feb 20th)
  - i. Internal testing results (Feb 27th)
  - j. User acceptance test results
    - i. Part 1 (April 3rd)
    - ii. Final (April 17th)
  - k. Abstract, introduction, implementation, results, discussion, conclusion (April 17th)
2. Final report
  - a. Increment 1 (March 6th)
  - b. Increment 2 (March 27th)
  - c. Final report (April 24th)
3. Code submissions
  - a. 1 (Feb 27th)
  - b. 2 (March 6th)
  - c. 3 (March 20th)
  - d. Fully working system (March 27th)
4. Documented code + GitHub submissions (April 24th)

Most of the tasks will be worked on collaboratively within the team to ensure everyone is on the same page with one another and with the customer. However, some of the explicit responsibilities are listed by team member below:

Prak: Risks & Challenges – Identification, Mitigation, Monitoring & Management

Sammy: Scope – Definition, Monitoring & Control

Wyatt: Quality – Definition, Monitoring & Control

Emmeline: Stakeholder Management / Communication

David: Schedule – Definition, Monitoring & Control

Additional Project Management Plans and Paradigms are outlined below:

Stakeholder Management and Monitoring Plan - plan for getting cooperation from stakeholders that are high power and high interest; this will also uncover influencers that are in the high power and low interest category, that have the ability to make or break a project.

- Meetings scheduled for every Thursday with customer at 3pm
- At these meetings, ensure that customer and development team both agree on which way the product should go
- Discover exact parameters for product features as early as possible
- Prepare explanations of each stage of our development process that can be understood by an audience with low-to-no technical experience

Risk Management Plan – plan for mitigation and monitoring of potential problems, then a plan for managing the risk to reduce the impact, should it become reality.

- Spend time each Sprint Review talking about bugs and issues discovered while working on implementations
- Communicate early and often with various stakeholders to prevent miscommunications or mismatched values, especially with Professor Wade and the customer

Quality Plan – plan for realizing the quality objectives as measured by the key performance metrics (KPM)

- Use user stories as KPMs in order to determine success of implementation
- Gather and present user stories early, with various iterations throughout the development process

Team Issue Log – lists critical issues that you want the team to focus on, as these may make / break your project

- Because the team has not yet begun design or implementation, we are currently not aware of any critical issues. Some expected difficulties of the project are:
  - Natural Language Processing implementation
  - Staying up to date with documentation and reports/project management

### **A1.3 Budget Costs**

Our sponsor has stated an intent to use Amazon Web Services for hosting. Pricing will be higher for a dynamic website than a static one, and it is likely that hosting will cost at least \$1,000 each year; our sponsor should expect this recurring cost to increase as the size of the user

base increases. For a more detailed breakdown of AWS pricing, our sponsor would have to speak directly with an Amazon representative.

This project should not require the purchase of any softwares, as many excellent IDEs are available for free, and the cost of Microsoft Sharepoint has been covered by the university. For the sake of accounting for all costs, the highest level of Office 365 costs \$23 per person per month, meaning that the cost of this software for 5 people from January to May will be \$575.

Development will require five computer systems capable of running web development software, on which to write, design, and test the product. The price of a new, professional Windows laptop with an Intel i7 core (not cutting-edge, but not egregiously out-of-date) from trusted retailers like HP or Dell tends to range from \$650 to \$1050 each. Depending on the team's exact needs and the timing of sales, the cost of computer systems could range from \$3250 to \$5250. This should be a one-time purchase. If the systems are purchased new, they are unlikely to require maintenance costs such as battery or charger replacements during the 15-week development period.

As expected for a 3 credit hour class, each team member will be working an estimated 12 hours per week on this project, to design, write, and test it. Since the average salary of a junior software developer in Texas is approximately \$42 per hour, our sponsor would expect to pay the team as a whole an estimated \$2520 per week for the duration of the project, which adds up to a total labor cost of \$37,800 for the full project.

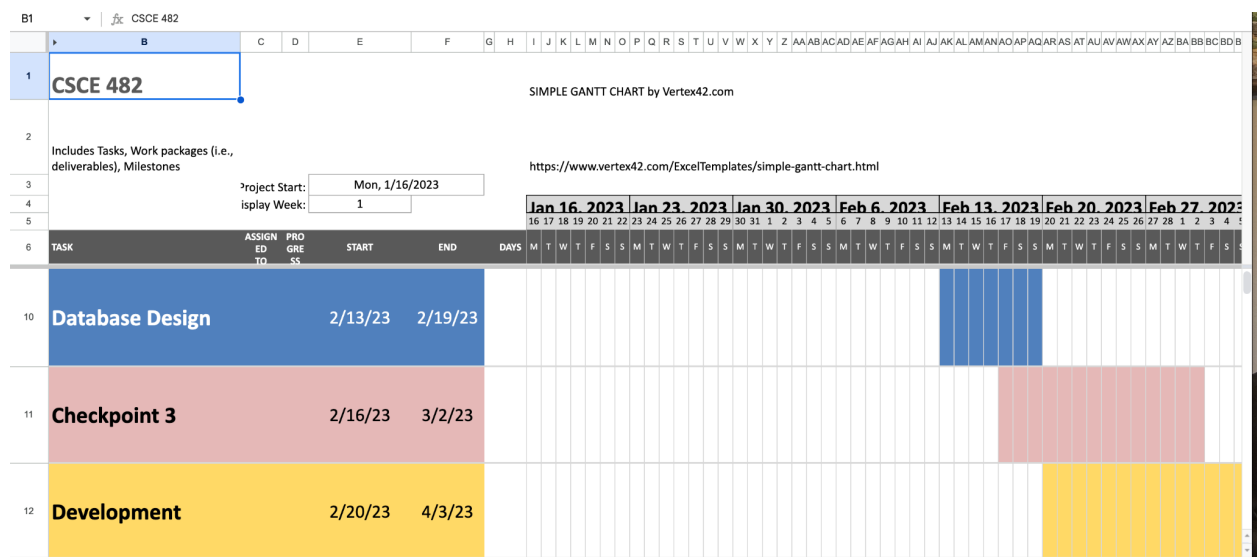
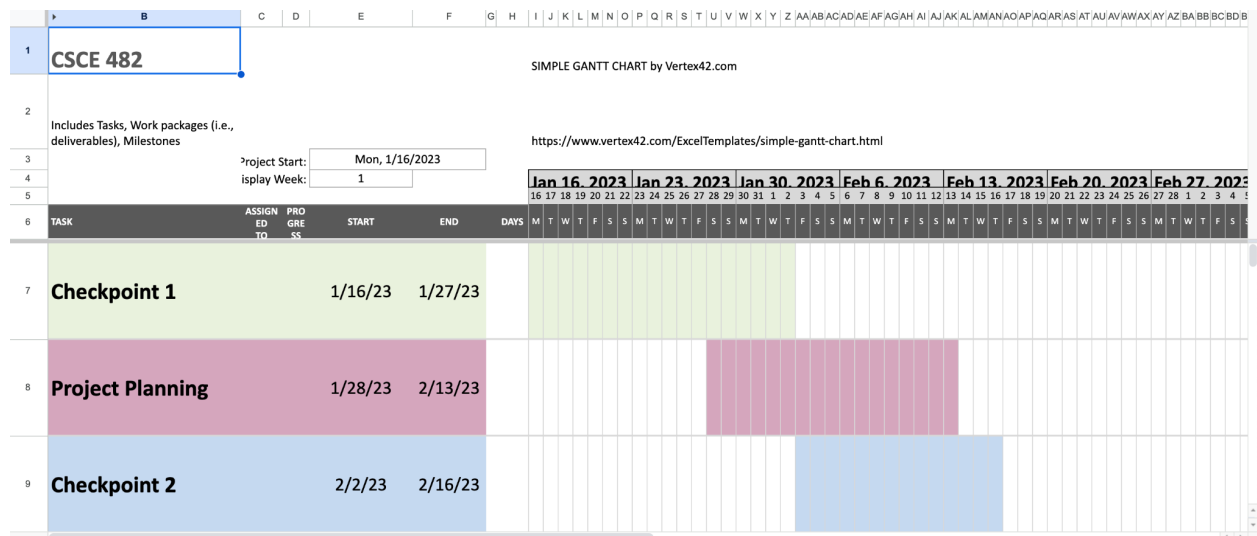
If our sponsor were hiring an external team for this project, they could expect to pay between \$41,625 and \$43,625 for all initial development costs, with yearly recurring costs of at least \$1,000 for Amazon Web Services hosting.

In reality, our team will receive no salary. We have already purchased five laptops capable of running web development software, as well as necessary peripherals such as chargers, and the only necessary expenditure will be our sponsor's purchase of web hosting, as well as Texas A&M University's purchase of Office 365. Therefore, the true budget for this project is \$1,575, for the initial purchase of Amazon Web Services hosting and the purchase of Microsoft Office 365. AWS hosting is a recurring cost that will increase yearly as the size of the product user base increases, but Office 365 can be safely canceled after the project has ended. Fortunately, Team Intelligent Child will not be responsible for these costs, since the price of hosting will be covered by our sponsor, and the price of Microsoft Office 365 has already been covered by Texas A&M.

## **Annex 2: Project Management Artifacts**

As part of our project, we have implemented an Agile process model to ensure efficient and effective project management. To keep track of our progress, we have created a Gantt chart that provides a comprehensive overview of our project timeline and milestones. Additionally, we have maintained an issue log to document any challenges or obstacles that we encounter during

the project. This log is regularly updated to ensure that we are addressing any issues in a timely and effective manner. To stay aligned as a team, we have also been submitting weekly status reports outlining our progress, challenges, and plans for the upcoming week. These reports serve as a valuable tool for communication and collaboration within the team, as well as keeping our stakeholders informed. We are committed to delivering a high-quality product and believe that our Agile process model, Gantt chart, issue log, and weekly status reports have been instrumental in ensuring our success.





[illegible]

## Annex 3: User Stories / Usage Scenarios

As part of our project, we have compiled a comprehensive list of user stories and usage scenarios. These documents outline the various ways in which our product will be used and the specific features and functionalities that our users will require.

By identifying and documenting these user stories and usage scenarios, we can ensure that our product meets the needs and expectations of our users. This information also serves as a valuable tool for our development team, as it guides the development process and helps prioritize features and functionalities.

1. As an expectant mother, I should be able to search for resources in my own language because I am not familiar with medical terminology.
2. As an expectant mother, I should only receive relevant resources from my search because I may not be able to parse out unwanted results.
3. As a system administrator, I should be able to add resources to the IntelligentChild database in case new resources become available.
4. As a system administrator, I should be able to edit resources in the IntelligentChild database in case existing resources deprecate, move, or otherwise change in a meaningful way.
5. As an expectant mother, I should only see a handful of curated resources because too many may overwhelm or confuse me.
6. As an expectant mother, I should be able to search/browse the application knowing that data is secure because the data being accessed can be quite sensitive and I may not want others knowing what I am searching.
7. As an expectant mother, I should be able to receive the information I need without it being unavailable, in case the information needed is critical.
8. As a system administrator, I should be able to identify if any existing resources are irrelevant and should be removed.

9. As a young expectant mother, I should be able to understand how the application works easily.
10. As a system administrator, I should be able to change the interface to make the application easier to use.
11. As an expectant mother, I should be able to search for financial aid for which I am eligible, because it would waste time to apply for resources for which I am not eligible.
12. As a system administrator, I should be able to add fields to resources in the database in case the medical landscape changes in a way that makes additional information necessary.
13. As an OB/GYN, nurse, midwife, or other medical professional, I should be able to refer my clients to the application, in case my client has a question when I am unavailable.
14. As an OB/GYN, nurse, midwife, or other medical professional, I should be able to request that my information, or my practice's information, be added to the database, so that expectant mothers in need can be more effectively directed to the resources they need.
15. As an OB/GYN, nurse, midwife, or other medical professional, I should be able to provide medical information from the resources the application provides to my client.
16. As a medical professional, I should be able to verify that the information the application provides to my client is factually correct and should be followed.
17. As a system administrator, I should be able to identify any searches that need to be fixed and extended further
18. As a system administrator, I should be able to encrypt and protect the data of all users in the IntelligentChild database/application.
19. As a provider, I want to ensure that the information provided to expectant mothers is only the most important and necessary information
20. As a young and expectant mother, the searches need to be executed quickly and not take very long

## Annex 4: Requirements Models

Weights add to 1 for each sub-category (i.e. All objectives in “Online/Easily Accessible” sum to 1)

<b>Online/Easily Accessible</b>	<b>Appealing Visual Design</b>	<b>Easily usable without search engine skills</b>	<b>Provide relevant and important information</b>
<b>0.3</b>	<b>0.1</b>	<b>0.3</b>	<b>0.3</b>
High Availability <b>(0.3)</b>	Intuitive Design <b>(0.3)</b>	Natural language processing <b>(0.5)</b>	Database with medical information <b>(0.5)</b>

Mobile browser compatible <b>(0.5)</b>	Matching thematically with the rest of the OliviaHealth suite <b>(0.5)</b>	Processes language with slang, misspelling, etc... <b>(0.5)</b>	Does not provide erroneous information (user may not be able to tell) <b>(0.35)</b>
Screen reader compatible <b>(0.1)</b>	Appealing to target demographic (young first time mothers) <b>(0.2)</b>		Prompts user to enter information absent in initial query that is necessary to provide best results <b>(0.15)</b>
Search engine optimized to appear high in Google rankings <b>(0.1)</b>			

## Annex 5: Prototype

Intelligent Child

Search the Intelligent Child database answers to your questions!

Results



Search the Intelligent Child database answers to your questions!

i am sick what should i do?

Results

## Your Results

Resource Name:

BCS Prenatal Clinic

Resource Description:

The Prenatal Clinic provides medical care during pregnancy. Some basic services provided include physical examinations, distribution of prenatal vitamins, and laboratory work. Other services may also be available; however, other services are provided on an as needed basis to be determined by the nurse practitioners and physician staff who volunteer at the clinic. The clinic offers a comprehensive prenatal program that includes: Prenatal Nutrition Physical/emotional changes during pregnancy and birth Prenatal and postpartum exercise Preparation for labor and birth Parenting skills Breast-feeding Newborn care and feeding

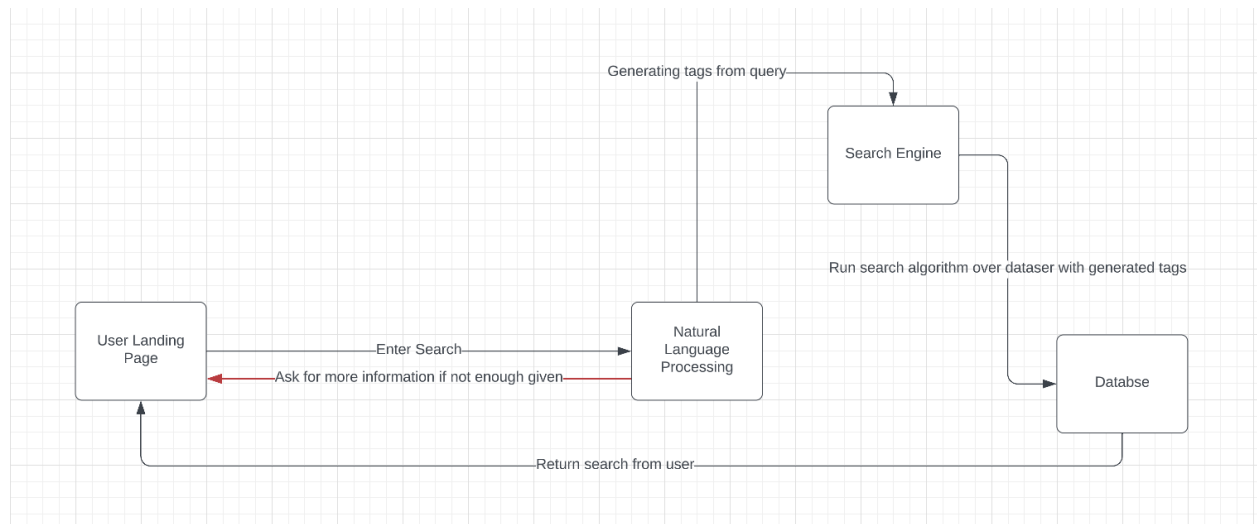
Resource Contact Information:

Email: BCSprenatal@gmail.com

# Annex 6: Design Models

```
_id: ObjectId('6420a681dd7ba3a6f34f8606')
Last Name: "City of College Station Community Development"
First Name: "City of College Station Community Development"
Title: "Community Resource"
Organization: "City of College Station Community Development"
Email: ""
Cell Phone: ""
Work Phone: "979-764-3778"
Role Rel'ps: ""
Description: "Down payment housing assistance and deposit assistance. The City of Co_"
TAMU Affiliation: "No"
Street Address: "1101 Texas Ave."
City: "College Station"
State: "TX"
Country: "United States"
ZIP: 77840
Days of Operations: "M-F"
Provider (M/F): ""
Community Resource/ State Resource: "Community Resource"
PDescription: "City of College Station Community Development-City of College Station _"
```

```
_id: ObjectId('6420a681dd7ba3a6f34f860d')
Last Name: "Pride Community Center"
First Name: "Pride Community Center"
Title: "Community Resource"
Organization: "Pride Community Center"
Email: ""
Cell Phone: ""
Work Phone: "979-217-1324"
Role Rel'ps: ""
Description: "Services for LGBTQ+
             Provides a safe place for persons of all sexual o_"
TAMU Affiliation: "No"
Street Address: ""
City: ""
State: ""
```



## Annex 7: Test Cases

Test Writer: Wyatt			
Test case name	NLP test #1	Test ID #	NLP-01
Description	Checks that the NLP model is able to properly encode the user query to compare to the corpus	Type	White box
Step #1	Action #1: User types in “medication for pain in abdomen” into search	Expected result: Query is encoded as per normal	
Step #2	Action #2: User types “Maternal care near me”	Expected result: Query is encoded as per normal	
Step #N	Action #3: User types in “SELECT * DROP TABLES;”	Expected result: Query is rejected for having invalid tokens, is not encoded, and user is prompted to try another query	
Step #N+1	Action #4: User query is blank	Expected result: Query fails to be encoded and user is prompted to type another query	

- Queries that are accepted and have at least one relevant resource have that resource returned to them

Test Writer: Wyatt			
Test case name	Database Search #1	Test ID #	DS-01
Description	Ensures that queries that are close to existing resource descriptions will have matches, and those that do not will not	Type	Black box

Step #1	Action #1: Query is “I’m a pregnant woman and need local healthcare resources”	Expected result: top result is BCS Prenatal Clinic	
Step #2	Action #2: Query is “I’m running out of food”	Expected result: top result is Brazos Church Pantry	
Step #3	Action #3: Query is “I’m having an emergency”	Expected result: top result is University Emergency Medical Services	
Step #4	Action #4: Query is “Is Popeye’s or Cane’s better?”	Expected result: Nothing is returned	

- Queries can have multiple responses or just one

Test Writer: Wyatt			
Test case name	Database Search #2	Test ID #	DS-02
Description	Ensures that queries that are close to existing resource descriptions will have matches, and those that do not will not	Type	Black box
Step #1	Action #1: Query is “I’m a pregnant woman and need local healthcare resources”	Expected result: BCS Prenatal Clinic, Elizabeth House Maternity Home, Women’s Care Plus, Brazos Valley Women’s Center and Health for All are returned	
Step #2	Action #2: Query is “I’m running out of food”	Expected result: Brazos Church Pantry, Brazos Valley Food Bank, Meals on Wheels are returned	
Step #3	Action #3: Query is “I’m having an emergency”	Expected result: University Emergency Medical Services	

Step #4	Action #4: Query is “Is Popeye’s or Cane’s better?”	Expected result: No results are returned	
---------	---	---	--

## Annex 8: User Acceptance Test Artifacts

The techniques employed for determining user acceptance involve the use of artifacts, which include a pre-survey, post-survey, and a survey that comprises both quantitative and qualitative components. Subsequently, user interviews will be conducted following the completion of the survey to gather additional insights from the participants.

### Survey

In order to get feedback after each participant performs their UAT, we will use Google Forms to collect responses to their questions. In this survey, there will be two distinct sections: quantitative analysis and qualitative analysis. This survey will be 15 questions long.

#### Quantitative:

- The first ten questions of the survey will be asking participants to rate certain aspects of the application on a scale of 1-10, with 1 representing a failure and 10 representing a success. Using these scores, we can track compliance with the specific goals of the project. Some example questions are listed below:
  - On a scale of 1-10, how intuitive was the application to use?
  - On a scale of 1-10, how relevant were the presented resources to your search?
  - On a scale of 1-10, how confident are you in the accuracy and validity of the resources provided?

#### Qualitative:

- The final five questions of the survey will be open ended/free response to allow each user to explain some of their thoughts while using this application. While this section will not be useful quantitative analysis, this will hopefully provide insight into the trends in the earlier quantitative section. Some example questions are listed below:
  - Was the website easy to navigate and use? Please explain if any part of using this application was difficult or confusing.

- Were the search results relevant to your question? If not, please give an example of a resource that was not relevant to the search you entered.
- Is there any other feedback you would like to give the team? Please enter any other opinions you have that were not asked in another question on this survey.

Due to the nature of the agile development process, we will likely not have every functionality implemented in the first UAT round. In this case, the survey will be cut down and the questions will be refined to match the MVP submitted for testing.

## **Interview**

The interviews for UAT will be minimal as the surveys will make up the bulk of feedback collected. However, we would still like to speak with the users who used the software as not all feedback can be encompassed within a survey. There will not be any specific questions for the interview section; rather, the team member administering the UAT will ask questions pertaining to the user's specific experience. This will be done *before* the survey is completed.

Example: A user opens the application and can't figure out how to run the search (search button isn't easily noticeable, confusing UI...). After the test, the administering member will ask about the specific difficulties around finding the search button.

# **Annex 9: Implementation**

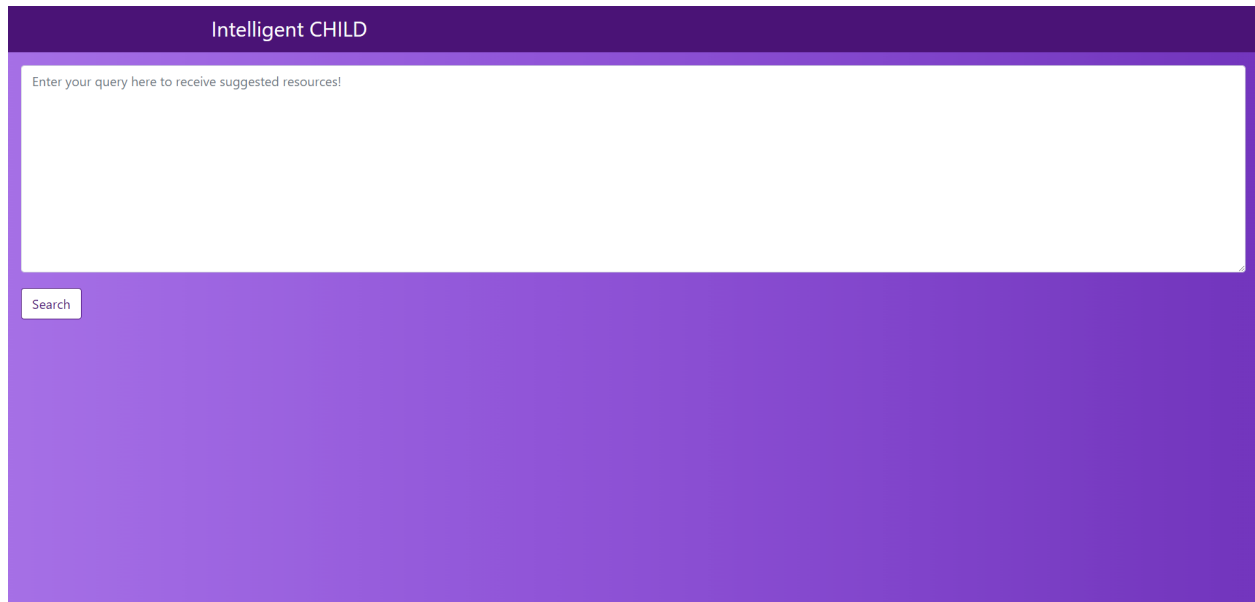


Figure 1: Screenshot of index page of website

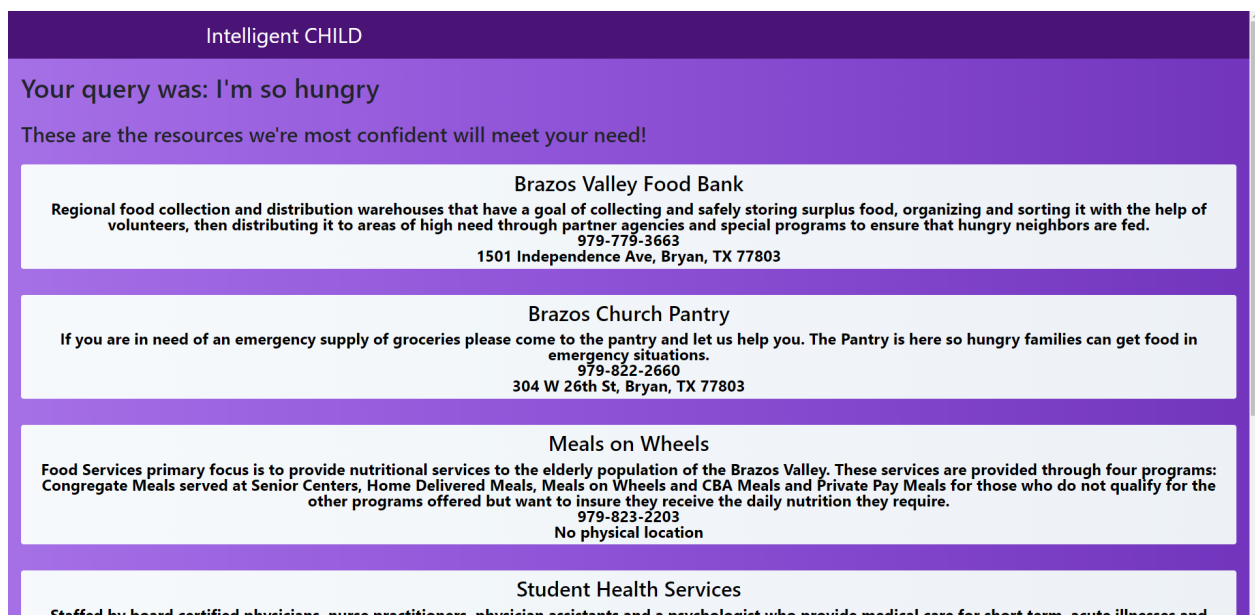


Figure 2: Response to user query

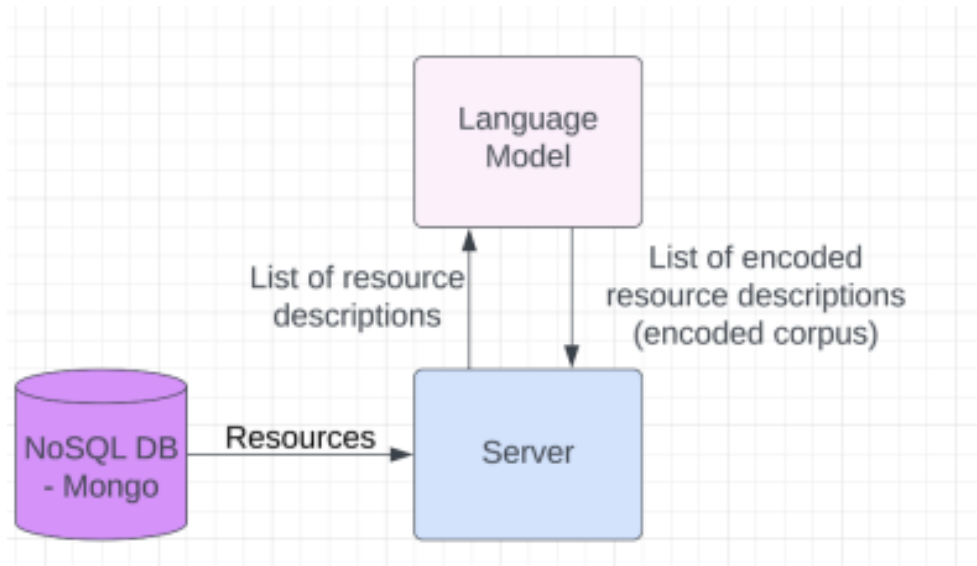


Figure 3: Encoding the corpus

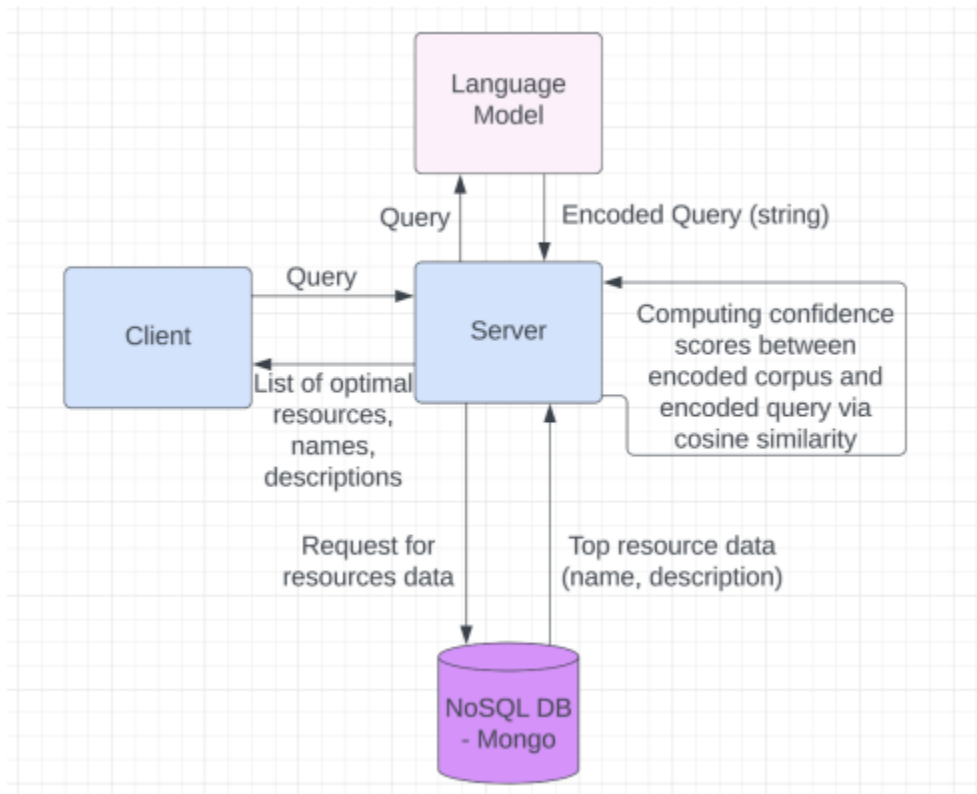


Figure 4: Encoded query and corpus comparison



1	I'm struggling with drinking too much and can't quit.	A drug and addiction treatment center to help fight addicton.	0.9
2	I need help quitting drugs.	A drug and addiction treatment center to help fight addicton.	0.9
3	Can you drink during pregnancy?	A drug and addiction treatment center to help fight addicton.	0.5
4	Where can I find an ultrasound?	A drug and addiction treatment center to help fight addicton.	0.2
5	Watermelons are on the side of the street after dusk.	A drug and addiction treatment center to help fight addicton.	0
6	Can prescription drugs be addictive?	A drug and addiction treatment center to help fight addicton.	0.7
7	I've been drunk every day for a month and need help.	A drug and addiction treatment center to help fight addicton.	0.9
8	Where can I find prenatal care?	Cover a multitude of practice areas including consumer, landl	0.2
9	I can't afford groceries right now.	Cover a multitude of practice areas including consumer, landl	0.2
10	I can't afford legal costs for a speeding ticket.	Cover a multitude of practice areas including consumer, landl	0.8
11	How can I sue someone for sexually assaulting me?	Cover a multitude of practice areas including consumer, landl	0.6
12	The bank is foreclosing my home and I need a lawyer.	Cover a multitude of practice areas including consumer, landl	0.7
13	Fishing is a very fun activity.	Cover a multitude of practice areas including consumer, landl	0
14	I need to find legal help for my boyfriend.	Cover a multitude of practice areas including consumer, landl	0.8
15	Where can I find help for my child with Down Syndrome?	Mental services for children and adults. Public non-profit com	0.8
16	I feel incredibly anxious and am worried about my mental health.	Mental services for children and adults. Public non-profit com	0.9

Figure 5: Sample of training data

## Annex 10: User's Manual

This particular application does not require any hardware or software installation on the part of the user. Instead, users only need to access the application through the provided URL, which is currently hosted at <http://wickedwyatt99.pythonanywhere.com/>. Upon accessing the landing page, users are presented with a search bar where they can enter any question of interest and subsequently retrieve relevant results. Once a user initiates a query, the application executes a search algorithm and provides a list of resources that correspond to the user's question.