

# Design Report for Chatbot Teaching Assistant

Sponsored by Lockheed Martin

April 27, 2022

## PREPARED BY

William Thramann	(Industrial Systems and Managerial Engineering)
Mitesh Kumar	(Computer Science / Computer Engineering)
Roland Rao	(Computer Science / Computer Engineering)
Sandor Magyar	(Computer Engineering / Computer Science)
Zachary Johnson	(Computer Engineering)
Leyang Zhang	(Computer Engineering)
Zhuoming Wu	(Electrical Engineering)
Jonah Que	(Computer Engineering)
Megan Stahl	(Computer Engineering)

PROJECT ENGINEER	Junichi Kanai (CORE)
------------------	----------------------

CHIEF ENGINEER	Nima Ahmadi (ISE)
----------------	-------------------

## Executive Summary

The Chatbot Teaching Assistant is a website where students of the course Capstone Design at Rensselaer Polytechnic Institute (RPI) can enter questions about the course in natural English and receive automatically generated responses. The answers are based on the Capstone Support Wiki, which contains most course information and is hosted on the main course website and project management platform known as the Electronic Design Notebook (EDN). The questions can be about anything on the wiki, including due dates or instructions for assignments like design presentations and reports, procedures for purchasing parts, or guides to using EDN itself.

This team is the team to work on the project, which was started by Capstone Design instructor Dr. Junichi Kanai to improve students' productivity and make them less likely to have to ask instructors for help. As an auxiliary benefit, instructors can analyze students' interactions with the bot to inform improvements to the Capstone Support Wiki or lesson plans.

Current and former students who responded to a survey distributed by the team agreed with Dr. Kanai, saying the wiki is difficult to navigate and that they would appreciate faster access to most information it contains. To improve the situation, the team has developed and deployed a website with two interfaces: Advanced Search and Rule-Based Chatbot.

Advanced Search passes the student's query to a machine learning model that selects the most relevant passage in the text of the wiki as an answer. Because the machine learning model matches synonyms and different sentence structures with similar meanings, Advanced Search returns more relevant results than the search feature built into the wiki. Answers are based on a database of pages downloaded from the Capstone Support Wiki daily, so Advanced Search stays up to date with edits to the wiki.

Rule-Based Chatbot matches its query against a decision tree manually constructed by the team and course instructors, taking users through a path in the tree and prompting for further information until an answer is reached. This system is faster and more reliable but less complete than Advanced Search, so students are advised to try Rule-Based Chatbot first. To continually improve Rule-Based Chatbot, anyone can add to the decision tree using a Web-based tool provided by the team, but only team members can put their changes into effect.

Team members alpha tested the system by using it to answer a list of queries based on the topics students needed the most help answering according to the user survey. Based on the initial tests, trying Rule-Based Chatbot, and then Advanced Search if it failed, took an average of 55 seconds (69%) less to arrive at a satisfactory answer than the search feature built into the wiki. The average F1 score, a standard accuracy measure for question answering systems, on these same queries was 0.81 out of 1 (higher is better). This met the team's semester goal.

Further testing based on the team's system evaluation plan, including beta testing with students not involved in the project's development, is recommended in order to identify areas for effective improvement. Instructor feedback also indicates that better instructions for using the Web-based rule tree editor are needed. The Rule-Based Chatbot should be further improved by expanding the rule tree, which is currently immature, and improving the search experience for assignments and class periods by integrating a work-in-progress module that allows searching by date and other assignment attributes. Finally, if the system becomes frequently used, the team recommends distributing the code to multiple hosts to decrease response times when multiple simultaneous users access the computationally expensive Advanced Search.

# Contents

Executive Summary .....	2
Contents .....	3
List of Figures .....	5
List of Tables .....	5
Revision History .....	6
Glossary .....	7
1 Introduction.....	9
2 Project & Semester Objectives .....	9
2.1 Long-Term Objectives .....	9
2.2 Current Semester Objectives .....	9
3 Engineering Tools and Methods .....	11
4 Technical Background .....	12
4.1 Types of Chatbots.....	12
4.2 Chatbot Deployment Logistics.....	13
5 Customer Needs and Engineering Design Requirements .....	13
6 Subsystem Design.....	15
6.1 Frontend .....	16
6.2 Advanced search .....	17
6.2.1 Concept Development.....	18
6.2.2 Text Pre-Processing .....	21
6.2.3 Inverted Index .....	22
6.2.4 BERT Question & Answer Model.....	24
6.3 Rule-based chatbot .....	25
6.3.1 System Design.....	25
6.3.2 Rule-Based Tree Construction .....	26
6.3.3 Assignment and Class Table Construction .....	27
6.3.4 Instructor Feedback on Rule Tree Editor Tool's Ease of Use .....	27
6.4 Web Scraping .....	28
6.4.1 Concept Selection .....	28
6.4.2 System Design .....	29
6.4.3 Wiki Scraping .....	29
6.4.4 Forum Scraping.....	30
6.4.5 User Manual.....	31
6.5 Data Analysis and Visualization .....	32
6.6 Database Logging and Categorization .....	33
7 System Evaluation .....	33
8 Significant Accomplishments and Open Issues.....	36
9 Conclusions and Recommendations .....	36
10 References.....	38
11 Appendix A: Initial Deliverables and Dates .....	40
12 Appendix B: Project Plan.....	41
13 Appendix C: System Evaluation Plan.....	44
13.1 User friendliness.....	44
13.2 Question answering accuracy .....	45
13.3 Resource efficiency .....	46

13.4	Load handling.....	46
14	Appendix D: Ethical and Professional Responsibilities .....	47
15	Appendix E: Instructor Comments on Rule Generation Tool .....	48

## List of Figures

Figure 2-1: F1 scores of best Q&A model vs. human performance on SQuAD datasets (reproduced from [1]) .....	10
Figure 3-1: Summary of engineering tools .....	12
Figure 5-1: Histogram of student responses to customer needs survey .....	14
Figure 6-1: Overall subsystem hierarchy .....	15
Figure 6-2: Advanced Search subsystem operation .....	17
Figure 6-3: Overview of Advanced Search subsystem .....	18
Figure 6-4: Parameters of BERT Base and BERT Large .....	20
Figure 6-5: Performance of BERT Large and BERT Base versus training set size .....	20
Figure 6-6: Preprocessing regular expression example .....	21
Figure 6-7: Named Entity Recognition using the Spacy Python library. ....	22
Figure 6-8: Inverted index example .....	22
Figure 6-9: Spatial Indexing demonstration .....	23
Figure 6-10: Sliding window algorithm illustration. ....	24
Figure 6-11: Application of thread-pool to parallelize application of BERT tasks. ....	24
Figure 6-12: Simplified rule specification tree .....	25
Figure 6-13: Rule specification tree example .....	26
Figure 6-14: Web scraper flow chart .....	28
Figure 6-15: HTML breakdown of an EDN wiki page .....	29
Figure 6-16: EDN forums initial page .....	30
Figure 6-17: EDN forums second page table .....	30
Figure 6-18: EDN forums thread .....	31
Figure 6-19: Visualization with student answers .....	32
Figure 12-1: Gantt chart .....	43

## List of Tables

Table 6-1: Document retrieval concept selection .....	18
Table 6-2: Question answering mechanism comparison .....	19
Table 6-3: BERT Large concept selection .....	20
Table 6-4: Scraper module concept selection .....	28
Table 7-1: Question answering accuracy .....	34
Table 7-2: Advanced Search response times .....	35
Table 7-3: Total time to get an answer using Chatbot TA vs. existing EDN search .....	35
Table 11-1: Initial deliverables and dates .....	40
Table 13-1: Average time for students to arrive at a useful answer using the chatbot. Common student queries are from surveys .....	44
Table 13-2: Average time for students to arrive at a useful answer by searching the wiki manually. Common student queries are from surveys .....	44
Table 13-3: Summary of question answering accuracy evaluation procedure .....	45

## Revision History

Version	Date	Name	Reason for Changes
1.0	1/27/22		Initial document
2.0	1/31/22	All	Statement of Work
3.0	2/24/22	All	Preliminary Design Report
3.1	4/14/22	All	Implement Dr. Ahmadi's suggestions on PDR
4.0	4/27/22	All	Final Design Review

## Glossary

Term	Definition
EDN	Electronic Design Notebook, the project management tool and course website for Multidisciplinary Capstone Design at Rensselaer
Rensselaer, RPI	Rensselaer Polytechnic Institute
F1 score (classification)	<p>A measure of the accuracy of binary classification models (higher is better).</p> $100 \times \frac{TP}{TP + \frac{FP + FN}{2}}$ <p>Where,</p> <p>TP = count of true prediction where true was expected,  FP = count of true prediction where false was expected,  FN = count of false prediction where true was expected</p>
F1 score (question answering)	<p>A measure of the accuracy of question answering models (higher is better).</p> $\frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$ <p>Where,</p> <p>Recall = the ratio of the number of shared words between the prediction and the correct answer to the total number of words in the correct answer,  Precision = the ratio of these shared words to the total number of words in the prediction</p>
HTTP request	Mechanism for requesting data from, or sending it to, a server over Hypertext Transfer Protocol (HTTP).
curl	A UNIX command-line tool for transferring data over network protocols like HTTP.
API	An application programming interface (API) is a set of functions, variables, data types, or other programming objects that form the interface by which other programs can interact with a specified program (for example, the API might provide functions for answering questions). An API can be made remotely accessible via network protocols like HTTP.
Memory, RAM	Random access memory. Main working memory space programs use to store data needed while running. Flushed upon system restart.
Disk	Stable storage that persists through system restart. Usually organized into files and directories, which programs like file explorers can access.

Reverse index, inverted index	Mapping of keywords to the sections of text that contain them. <i>Inverted</i> because the plain text itself is a mapping from sections of text to the words they contain.
Spatial index	Mapping of keys to points in space to allow for similarity comparison by spatial locality.
Docker container	Lightweight, standalone, executable package of software that includes everything needed to run an application.
UNIX process	A running instance of a program managed by the operating system, with independent virtualized memory and potentially the ability to run simultaneously with other processes.



# 1 Introduction

The goal of the Chatbot Teaching Assistant project is to develop and deploy a chatbot that helps future students in Multidisciplinary Capstone Design at Rensselaer Polytechnic Institute (RPI) find assignment details, part purchasing guidelines, project management procedures, and other information about the course. This course information is already in a wiki called the Capstone Support Wiki on the course website (the *Electronic Design Notebook*, EDN), but it is not always easy to find, especially because the course is large and diverse and involves the whole engineering design process. This project is a process improvement activity guided by Dr. Junichi Kanai—one of the study sponsors—to facilitate student productivity and reduce the load of questions on instructors by giving students better command of the wiki. The project is sponsored by Dr. Kanai and Lockheed Martin.

The project's users are future Capstone Design students and faculty. The students need answers to questions asked in plain English faster than searching the wiki, and the faculty need an easy way to update the system when the wiki changes and possibly view a log of students' questions to guide their instruction.

## 2 Project & Semester Objectives

This section details the team's expected accomplishments this semester, as well as over-arching goals of the project that should be the basis future work. As this is the first team to work on the project, the long-term objectives are nascent and should most likely be modified in the future, based on instructor feedback and ideas from future teams.

### 2.1 Long-Term Objectives

In the long-term, the project aims to create an automated assistant for Capstone Design at RPI. The overall objectives of this assistant are to help reduce students' confusion and make them less likely to have to ask instructors for help. The assistant keeps itself up to date with changes to the course and provides statistics to instructors to help them make relevant information more accessible to students.

- a. Have functional assistant to meet students' course information needs
- b. Save instructors' time by reducing the number of student questions to answer
- c. Easily adaptable to changes in the course, including the Capstone Support Wiki
- d. Data analytics provide meaningful feedback to instructors

### 2.2 Current Semester Objectives

The team's objective for this semester was to develop an application that analyzes the text on the EDN Capstone Support wiki to answer Capstone Design students' questions, gives instructors utilities for collecting diagnostic data from users, and keeps up to date with wiki changes. The team expects the system to handle at most one hundred simultaneous users, and updates to the instructor dashboard every ten minutes. This constraint was determined by counting the average number of students in a Capstone Design section, and the number of simultaneous meetings.

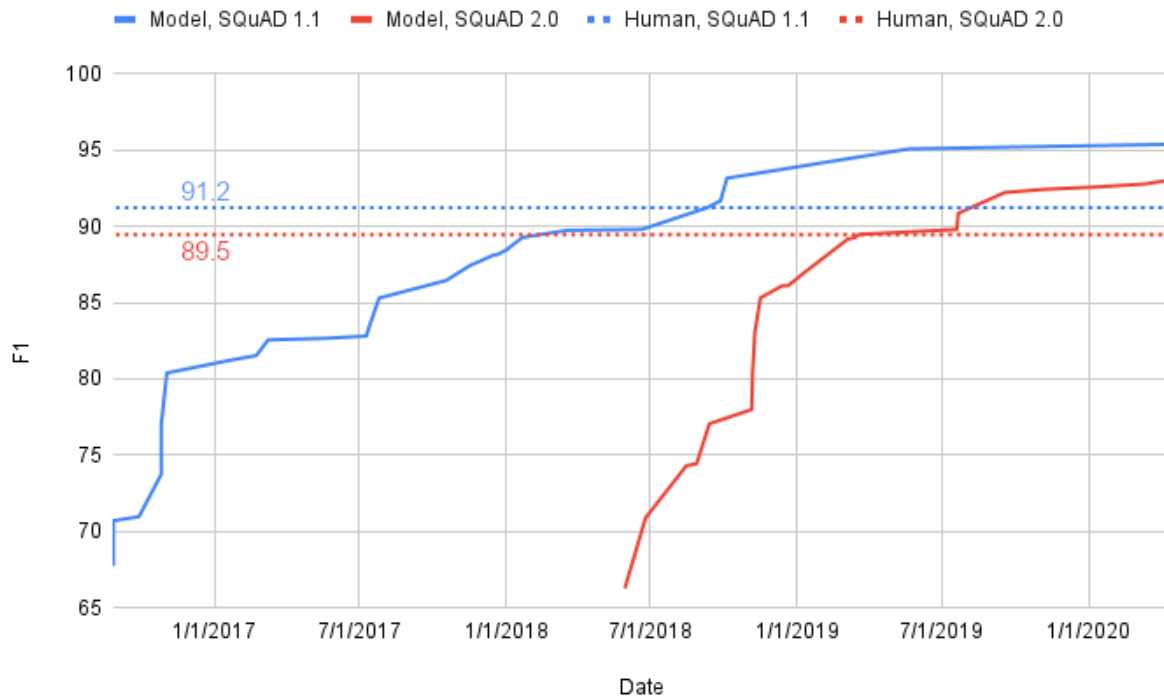
The application runs on a server and uses clients' machines solely to display the user interface, as running expensive computations on the client may increase response times. The performance of the system is evaluated by measuring the time it takes to arrive at a useful response and how

that compares with the existing EDN navigation capabilities. The specific evaluation plan is described in Appendix C: System Evaluation Plan and involves comparing the time it takes students to find the answer to a question using the chatbot to manual searching.

One important goal for the system is convenience, measured by the following parameters:

1. Average response time per user action
2. Number of relevant pages generated
3. Average time a student takes to get a satisfactory answer

Another desired characteristic of the chatbot is high accuracy; this is evaluated by how often the bot correctly classifies questions as answerable or un-answerable and how often the answerable questions get a satisfactory response.



**Figure 2-1: F1 scores of best Q&A model vs. human performance on SQuAD datasets (reproduced from [1])**

Figure 2-1 compares the state-of-the-art question answering machine learning models over time with human performance, as tested on the Stanford Question and Answer Dataset (SQuAD). SQuAD is a set of Wikipedia articles with reading comprehension questions and answers based on them. The human or computer model is given the articles and then quizzed with the questions.

The team set constraints for evaluating the chatbot based on this study. Accuracy of the question answering subsystem is measured using the F1 score (for question answering), which is defined in the Glossary. The state of the art in July 2018 achieved a 70% F1 score, as shown in Figure 2-1. Due to the time constraints of the semester, the team adopted this score of 70% as a target for evaluating the accuracy of the system.

Other future work out of the scope of this semester would focus on building user-specific experiences by using new data generated by project teams to deliver more relevant answers for a wider scope of queries. Privacy concerns with intellectual property that project teams will

generate make this challenging. Lastly, future efforts would go towards building custom models on real data from past students and faculty to further improve detail in the bot's responses.

In-scope deliverables:

1. Working trivial chatbot hosted on a server that can produce a response to some user interaction, such as a "hello" message, to verify that the server, connection, and frontend are all working
2. The EDN assistant is accessible through chat messages
3. Means for instructor to update chatbot database
4. Bot that can answer self-contained questions

Out-of-scope deliverables:

1. Bot that can ask for more info to answer more complex queries
2. Use user-specific EDN data outside of the wiki to tailor responses to each project team
3. Build custom models on data from past students and faculty to improve response detail

### 3 Engineering Tools and Methods

Various frameworks, programming languages, and libraries were used in the development of the system. The project is divided into three main areas of development:

1. Frontend
2. Information gathering
3. Question answering and analytics

The set of tools employed in each of the three sections vary greatly. They are itemized for each respective section below.

Frontend:

- **JavaScript:** Programming language for frontend development
- **React.js:** Framework for writing applications that run in the Web browser
- **Virtual machines:** Used for serving the applications on the RPI network

Information gathering:

- **Python:** Programming language for simple and versatile scripting

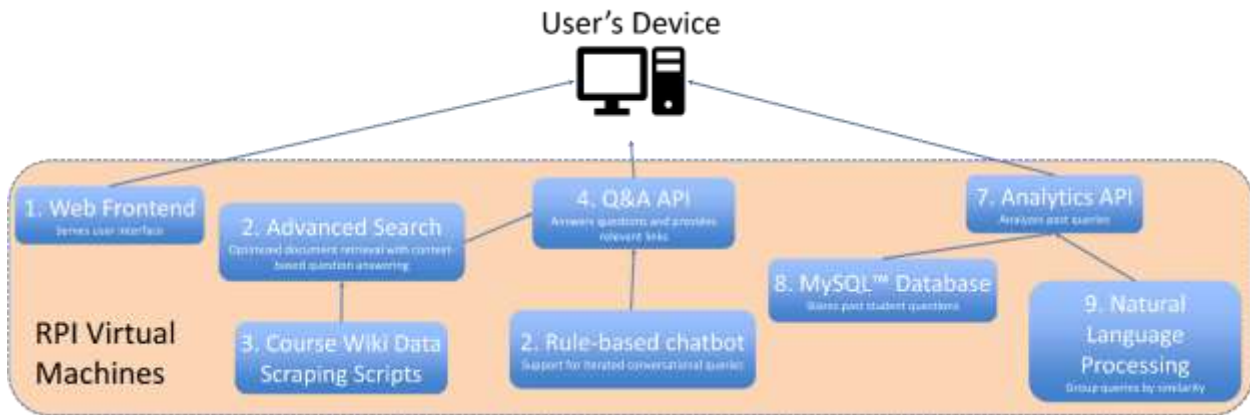
Question answering and analytics:

- **Python:** Programming language with extensive libraries for machine learning applications
- **Virtual machines:** Used for serving the question answering APIs to the frontend
- **SQL:** Querying language used to store and retrieve data from databases

Methods are ideas and concepts used to plan and execute the project. The methods the team used can be found below:

- **Customer surveys** to identify the most frequently asked questions to focus on
- **Customer feedback** to identify areas for improvement
- **REST API design** to simplify logic and decrease chance of error
- **HTTP requests** to enable communication between backend APIs and the user's device

- **Natural language processing** to process wiki data and user queries into a format useful to a computer program
- **Databases** to efficiently store and retrieve wiki data



**Figure 3-1: Summary of engineering tools**

Figure 3-1 shows the broad structure of the project. The user's device fetches the browser application code from the frontend Web server and uses it to access the Q&A and analytics APIs. The Q&A API consists of two user interfaces: Advanced Search and Rule-Based Chatbot. Advanced Search intends to help answer questions by directly scanning and understanding the EDN wiki. Rule-Based Chatbot provides a framework that allows instructors to encode potential avenues of conversation that the chatbot should be able to take depending on what the user asks. Advanced Search also depends on the information gathering system to build its knowledge base, from which it finds answers to user queries with natural language processing techniques.

Lastly, the analytics API collects logs from requests sent to the Q&A API and produces valuable diagnostic information to the administrators of the system. This makes use of SQL for persistence and natural language processing for evaluating similarity between bodies of text. The APIs follow the REST API design, meaning they do not maintain state between requests from the user. They are also accessible over the RPI network via the HTTP network request protocol. REST and HTTP are described in the Glossary and are commonly used in software development projects. The broad structure above, and additional libraries and tools selected after going through concept generation and selection processes, is explained extensively in Section 6.

## 4 Technical Background

Academic researchers have divided chatbots into several categories based on how they store data and produce answers. Each category is fundamentally different, so choosing between them was a crucial step for the team. Another choice the team made based on background research is the method of deployment for the chatbot. These two decisions are discussed in Sections 4.1 and 4.2.

### 4.1 Types of Chatbots

Chatbots have previously been applied to areas like customer service, school application information, and tax advice. For example, InGenious AI and Mi-fi Accountants & Advisors' TaxBot asks users a series of questions over Facebook Messenger to help them file their tax

returns [2]. TaxBot takes care of the logic of the tax return forms, so users need not read and understand their detailed instructions.

*Rule-based* bots like TaxBot present an interface much like filling out an online form: a linear sequence of questions that together collect all the information needed to file the tax return, answer a question, or provide some other help. Internally, the sequence of questions is not linear: the user's responses can determine which questions are asked next, and this logic is necessarily pre-defined and programmed into the bot [3]. The benefit is that every interaction programmed into the bot satisfies the user to exactly the extent the developers make it.

Besides rule-based, two other categories of chatbots are *generative* and *retrieval-based* [4]. *Generative* bots use machine learning to find and formulate answers from a knowledge base and even simulate conversation, while *retrieval-based* bots offer a level of flexibility between generative and rule-based by looking up information from a knowledge base but in predefined ways [5].

A strictly rule- or retrieval-based chatbot does not suit the team's needs because the team would have to manually program every kind of user query the bot is intended to support. Even then, the bot would be useless when asked unknown queries. However, rule- and retrieval-based models are more reliable for known questions. To maximize the potential of the information available on the wiki, a few rules for common questions backed by a basic generative model for everything else are best.

## 4.2 Chatbot Deployment Logistics

The server that accepts incoming user messages can be run directly by the team or via a third party like Cisco Webex. The third party might provide the user interface, security features, or other desirable services. If the user's computer communicates with the team's server directly, the team must design and implement its own user interface.

In any case, some computer has to communicate with the team's server. This can happen over two different protocols: HTTP and WebSocket. HTTP is how webpages are served over the Internet, and it works by a computer sending requests to the IP address of the team's server and receiving responses [6]. To prevent anyone on the Internet from doing this, whoever controls the team's server might restrict it to only accept requests from computers on RPI's network.

Since third parties like Webex are not on RPI's network, such a restriction would eliminate the third-party option and require students to connect to RP's VPN or be in range of campus Wi-Fi to use the chatbot. Before pursuing the third-party option, the team should ensure the chatbot's long-term host server will be publicly accessible.

WebSocket is an alternative to HTTP where the connection between the two computers is more direct and persistent and wouldn't require public access to the team's server, allowing the bot to work under network access restrictions [7]. This may also make it more efficient than HTTP [8], but this claim must be tested in the team's specific situation.

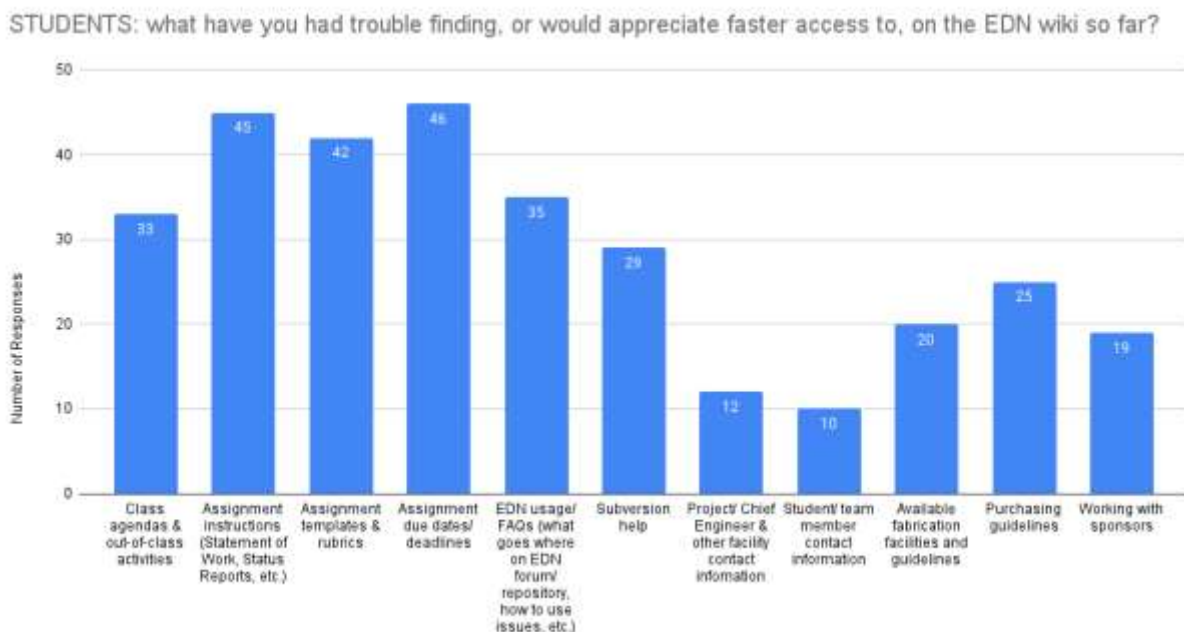
## 5 Customer Needs and Engineering Design Requirements

This is a summary of the project's Needs and Requirements Workbook [9] as it appears at Revision 47 of the team's Subversion repository under Background Information. In this section, each customer need and engineering requirement in the Needs and Requirements Notebook is

referenced by its number in parentheses. For example, (5) refers to Customer Need 5, “Chatbot needs information from wiki pages,” and (5.1) refers to Engineering Requirement 5.1, “Chatbot needs to be able to scrape information off of the wiki and store it in a database in order to be indexed,” which is designed to meet Customer Need 5.

The chatbot uses information from the Capstone Support Wiki (5), so it must scrape the wiki and store the information in a database (5.1). Since the bot would be useless if it were slower than only using the wiki (3), users should be able to arrive at a satisfactory answer within a minute (3.1). The chatbot must also be easy to use with natural English queries (1), which entails accurately extracting and responding to their meaning. Therefore, the team set the requirements that the system correctly determine whether 80% of queries are answerable based on the information in the wiki (1.1) and give a satisfactory answer to 50% (1.2).

A secondary purpose of the bot is to give instructors insight into student needs (2). The team’s goal for this is to provide a log of student interactions and some basic analytics over them (2.3).



**Figure 5-1: Histogram of student responses to customer needs survey**

The parts of the Capstone Support Wiki that needed the most attention were identified by creating a survey [10], which was posted on Reddit and sent out to current Capstone students. Students were asked to select the types of information they had trouble locating, and also to rate the convenience of a chatbot deployed through Webex versus a standalone webpage. As shown in Figure 5-1, respondents needed the most help finding assignment instructions, templates, rubrics, and due dates, as well as class agendas, homework, and EDN usage guides (7). These high-priority questions should be included in the hardcoded rule-/retrieval-based answer tree to ensure they get high-quality responses (7.1).



## 6 Subsystem Design

The subsystem hierarchy in Figure 6-1 defines the main components and their interconnectivity that constitute the end deliverable of this semester. As explained in Section 4.1, the team originally decided to design the Q&A subsystem to first try to match incoming queries with a set of predefined rules and then fall back to a machine learning model if no good match exists. This hybrid approach covers a wide range of queries with the machine learning model while leaving room for fine-tuning with rules, forming a comprehensive solution. However, rather than integrate the rule-based model (Section 6.3 Rule-based chatbot) with the machine learning model (Section 6.2 Advanced search), the team found it more valuable to use the remainder of the semester to bring each model separately to a higher level.

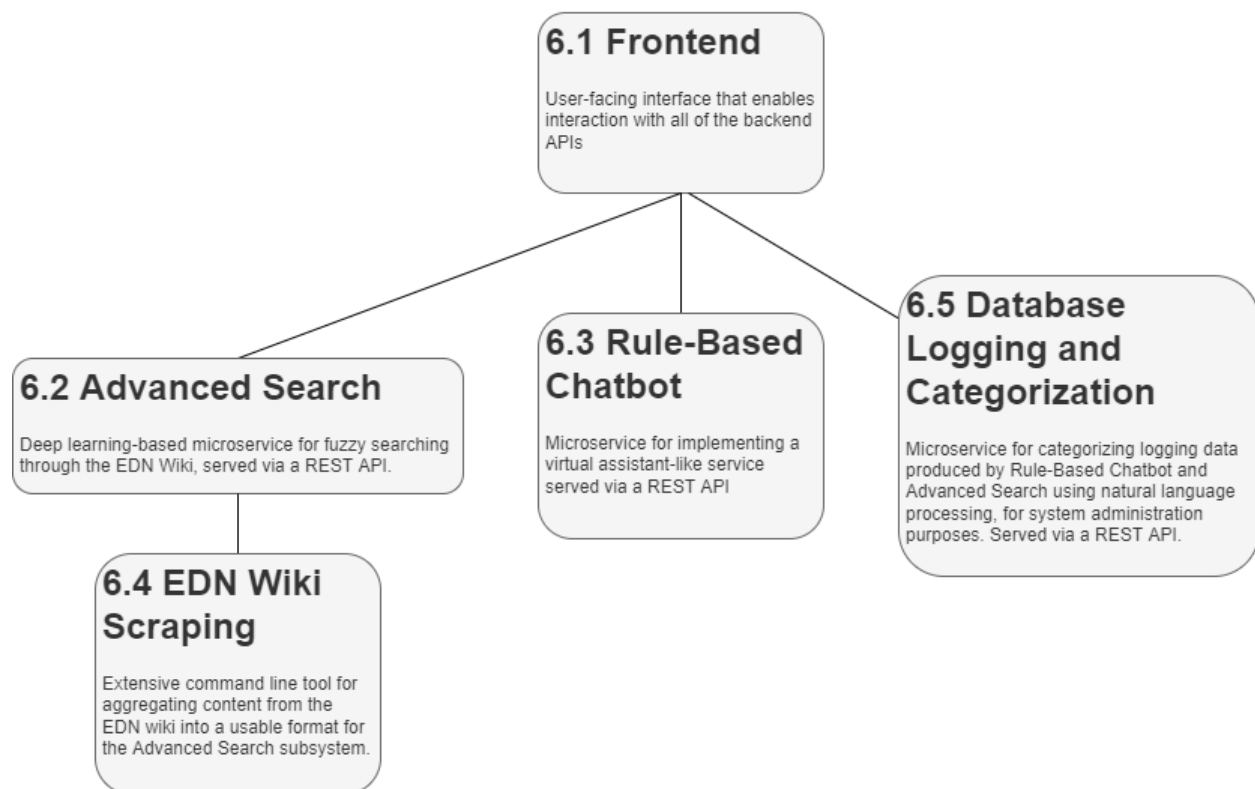


Figure 6-1: Overall subsystem hierarchy

The frontend (see Section 6.1) is served to the user as a Web application and provides a streamlined interface to the remote APIs: Advanced Search (Section 6.2), the Rule-Based Chatbot (Section 6.3), and Database Logging and Categorization (Section 6.6).

The Advanced Search API combines fuzzy document retrieval using spatial indexing with the context-based question answering capabilities of the BERT natural language processing architecture to create a powerful search tool. The Rule-Based Chatbot API provides a general framework for instructors to formulate their own rules and then serve them to the student as a virtual assistant. The shared goal of Advanced Search and Rule-Based Chatbot is to help students find information with minimal instructor involvement. These dual Q&A systems offer flexibility on the instructor's end for improving the system's user friendliness, as well as a higher likelihood of the user eventually finding their answer.

The Database Logging and Categorization API collects diagnostic data from Rule-Based Chatbot and Advanced Search interactions for use by instructors to improve the student experience by manually editing the wiki or Rule-Based Chatbot's rule tree, the knowledge bases of the two Q&A APIs.

Finally, the EDN wiki scraping (Section 6.4) command-line tool automatically collects plain text data from the EDN wiki into a usable format for the Advanced Search knowledge base. Each subsystem is described in further detail in its respective section below.

## 6.1 Frontend

The use of the React JavaScript framework was intended to reduce the time to develop a fully functional and modern-looking frontend. This subsystem is the most connected to different parts of the application through API requests for data transfer.

The home page features a chat interface that allows users to interact with a decision tree model that takes in multiple queries and directs the user to the answer that they are looking for. Another option on this page is the Advanced Search feature, which is a significant improvement upon the EDN wiki's own search, as it includes fuzzy searching for misspelled or synonymous words and features relevance sorting instead of chronological sorting.

As both of these functions run, queries are stored and displayed in the Instructor Dashboard. This page is meant to be viewed by instructors only and displays a log of all queries sent to Rule-Based Chatbot as well as all questions asked in Advanced Search. Categorization of queries is another table that is displayed in the frontend and is intended for use with query analytics. This feature will allow instructors to determine which areas have the least amount of information available to students and use that to improve the knowledge base for those sections.

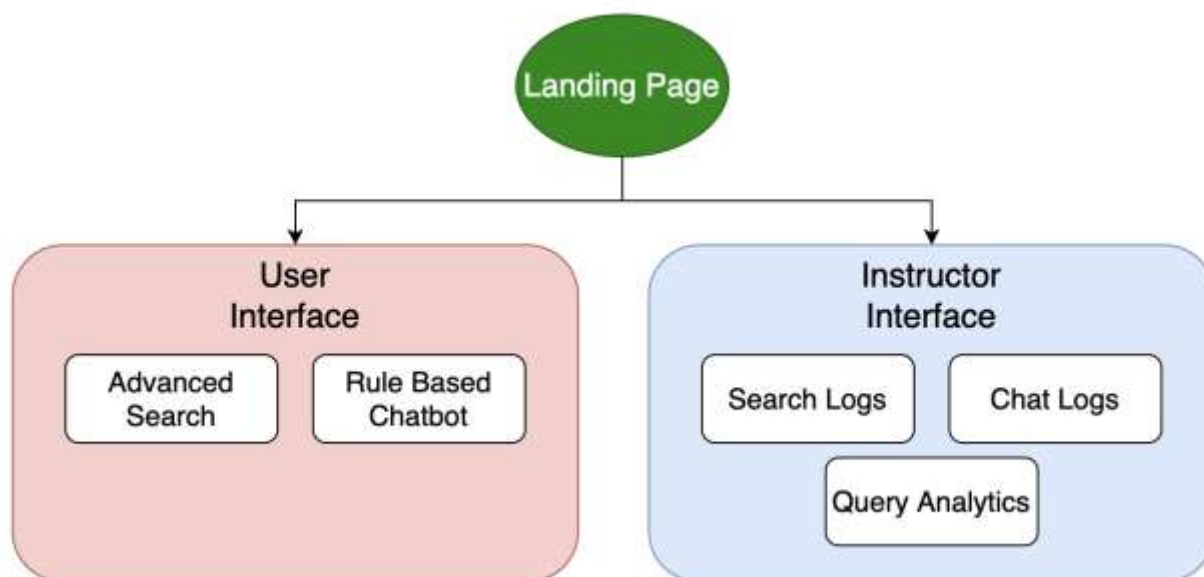


Figure 6-1: Frontend system architecture



## 6.2 Advanced search

Figure 6-2 shows this subsystem in action. The user's question is processed, relevant documents are retrieved, and the most relevant part of the text is highlighted. Despite the documents not containing the exact keywords that appear in the user's question, they are still detected as relevant due to the team's fuzzy search design, which will be elaborated upon next.

Query  
How do I manufacture or purchase a part?

Send

**\_PE-Purchasing\_**  
we do not need to purchase parts to see "if they work"  
e than just a concept! Teams need a concrete design documented that fully shows how the items will be used (schematic, CAD, etc.) as well as supporting analysis showing how/why the item will work within the overall system. In terms of the design process, this usually means passing through concept generation and selection and getting into detailed design - before ordering parts, especially expensive ones. In general, we do not need to purchase parts to see "if they work". Commercial parts are sold to meet th

[Web Link](#)

**RPI Fabrication Purchasing**  
sign purchase form and initial drawings  
of Design Lab). Complete the Purchase Form with all necessary information including an approximate cost for all items/services that will be utilized and the contact information of the person in charge of the respective campus service. Have PE review design and drawings - sign purchase form and initial drawings. If you have not done so, obtain approval signature from Sam Chiappone (JEC 3100A) for on or off campus manufacturing, rapid prototyping, machine shop, or physical facilities services. Work done on c

[Web Link](#)

**Purchasing\_Guidelines#Support-Services-On-or-Off-Campus-Ex-Machine-shop-3D-printing-rapid-prototyping-physical-facilities-Computer-Store-etc**  
fabrication area  
be either given to your Project Engineer who will deliver it to you or it will be put on your team bench in the fabrication area. The best way to contact me is via email (mastes@pi.edu). I check my email often so I will get back to you quickly. Unlike IED where students pay for all project materials, your Capstone project covers the costs for your project materials. The only anticipated team expense is for the printing poster at midterm and end of semester. As a result, it is critical that you follow the

[Web Link](#)

Query  
How do I buy or build a part?

Send

**\_PE-Purchasing\_**  
we do not need to purchase parts to see "if they work"  
e than just a concept! Teams need a concrete design documented that fully shows how the items will be used (schematic, CAD, etc.) as well as supporting analysis showing how/why the item will work within the overall system. In terms of the design process, this usually means passing through concept generation and selection and getting into detailed design - before ordering parts, especially expensive ones. In general, we do not need to purchase parts to see "if they work". Commercial parts are sold to meet th

[Web Link](#)

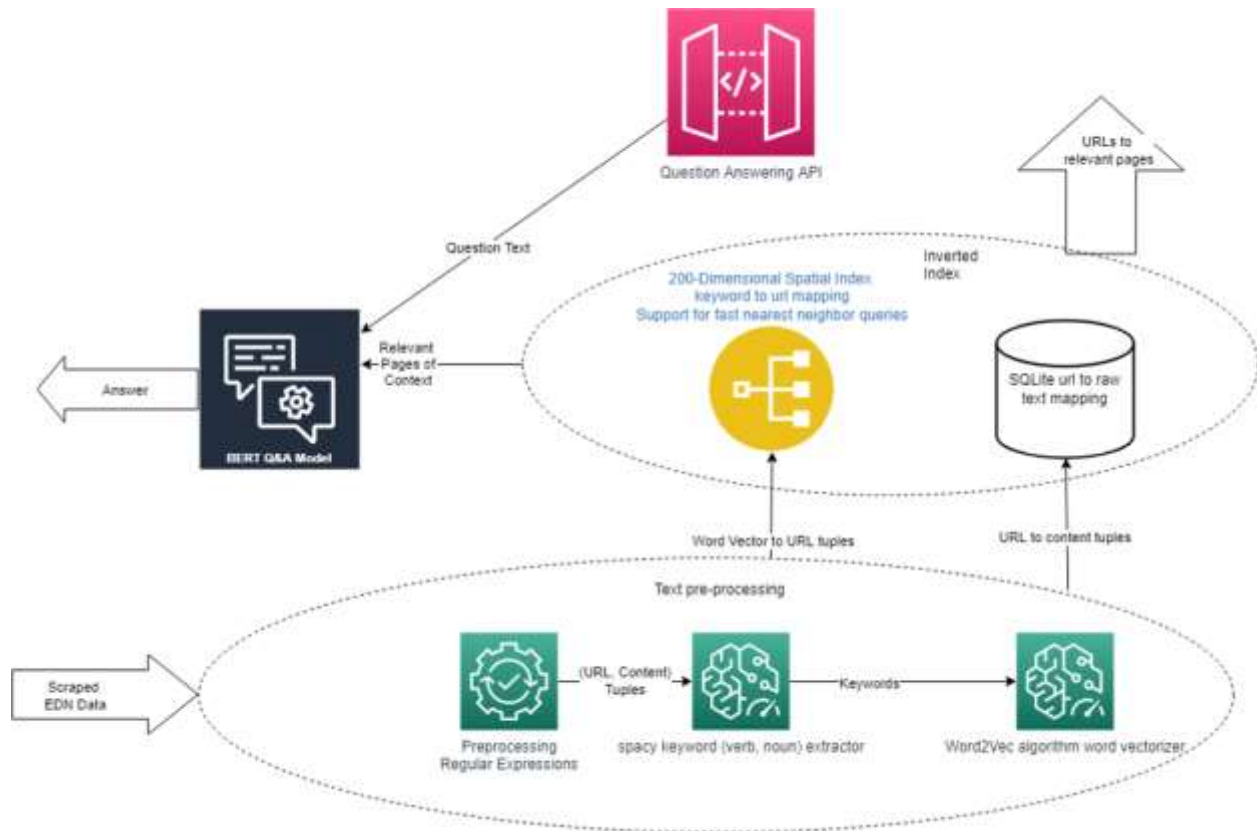
**Project Engineers-Purchasing**  
we do not need to purchase parts to see "if they work"  
e than just a concept! Teams need a concrete design documented that fully shows how the items will be used (schematic, CAD, etc.) as well as supporting analysis showing how/why the item will work within the overall system. In terms of the design process, this usually means passing through concept generation and selection and getting into detailed design - before ordering parts, especially expensive ones. In general, we do not need to purchase parts to see "if they work". Commercial parts are sold to meet th

[Web Link](#)

**PSoC**  
How ones may be purchased / obtained if needed  
Knowledge Base Technology Related The PSoC is a family of devices made by Cypress (http://www.cypress.com/) who has graciously sponsored projects in our lab. PSoC stands for "Programmable System on a Chip". We have a small set of development kits for these. See Booty Bay to reserve one for your project. New ones may be purchased / obtained if needed. While many teams worry about the physical size of the microprocessor they use, it should be remembered that you are typically using an evaluation or ready

[Web Link](#)

Figure 6-2: Advanced Search subsystem operation



**Figure 6-3: Overview of Advanced Search subsystem.**

Figure 6-3 contains the following three sub-components that work together to support the advanced search capabilities of the virtual assistant: text pre-processing, inverted index, and the BERT Q&A model.

The text pre-processing pipeline is used both in preparing the scraped EDN data as well as individual user questions sent to the API, for the spatial locality properties of the inverted index. The inverted index utilizes spatial locality to retrieve the most relevant documents in the scraped EDN data, by the similarity of their keywords to those in the user's question. These relevant documents are then fed into the BERT question answering model along with the user's question, which then produces a highlighted section in the text that best answers the user's question for each relevant document. The concept development as well as the final design of each element is described in more detail in the following items.

### 6.2.1 Concept Development

**Table 6-1: Document retrieval concept selection**

Concept Name	Concept Description	Ease of Implementation	Performance	Resource efficiency	Total	Select (Y/N)
Whoosh™ Database Inverted Index	1. Whoosh™ Database 2. Inverted Index	10	4	10	24	N

Elasticsearch™ Inverted Index	1. Elasticsearch™ Database 2. Inverted Index	10	6	10	26	N
(Word2Vec + Spatial Index) Inverted Index	1. Word2Vec Algorithm 2. Spatial Index k-nearest neighbor queries 3. Inverted Index	8	10	10	28	Y

Table 6-1 lists the different concepts that were considered when implementing the document retrieval system, and ultimately the Word2Vec algorithm in combination with spatial indexing to construct an inverted index was chosen. The team found that although it was lacking in ease of implementation, it had the highest performance as well as resource efficiency out of the other options. After testing against the Whoosh Database Inverted Index, the team concluded that the Word2Vec + Spatial Index Inverted Index concept produced the best results. Elasticsearch was not investigated as thoroughly due to its similarity to Whoosh, as well as its low accessibility due to it being a paid service.

**Table 6-2: Question answering mechanism comparison**

Concept Name	Concept Description	Ease of Implementation	Performance	Resource efficiency	Total	Select (Y/N)
Rule-/retrieval-based	Every possible interaction programmed into chatbot	9	2	10	21	N
Machine learning-based	Machine learning model interprets and answers questions	4	8	4	16	N
Rule-/retrieval-based with machine learning fallback	Only common questions, those requiring multiple message exchanges, and those machine learning fails to answer are programmed in as rules	4	10	8	22	Y

The next table, Table 6-2, lists out the factors that led the team to implement a rule retrieval based chatbot with a document retrieval question answering system as a fall back, as opposed to implementing one or the other. The rule retrieval-based approach scored high in ease of implementation and resource efficiency but was not general enough to be performant. The machine learning based document retrieval was only better for performance but lacked resource efficiency and ease of implementation. The team found that combining the two significantly improved performance by being the most comprehensive system for answering questions.

The natural language processing tasks that are key to the product are context-based question answering and named-entity recognition, and the team found that BERT large was the highest performing solution for these applications. BERT comes in two variants with different numbers of parameters. Figure 6-4 shows the total number of parameters in the respective architecture variants. BERT large is clearly significantly larger and therefore more resource intensive to train and operate.

- $BERT_{BASE}$  : L=12, H=768, A=12, Total Parameters=110M
- $BERT_{LARGE}$  : L=24, H=1024, A=16, Total Parameters=340M

Figure 6-4: Parameters of BERT Base and BERT Large

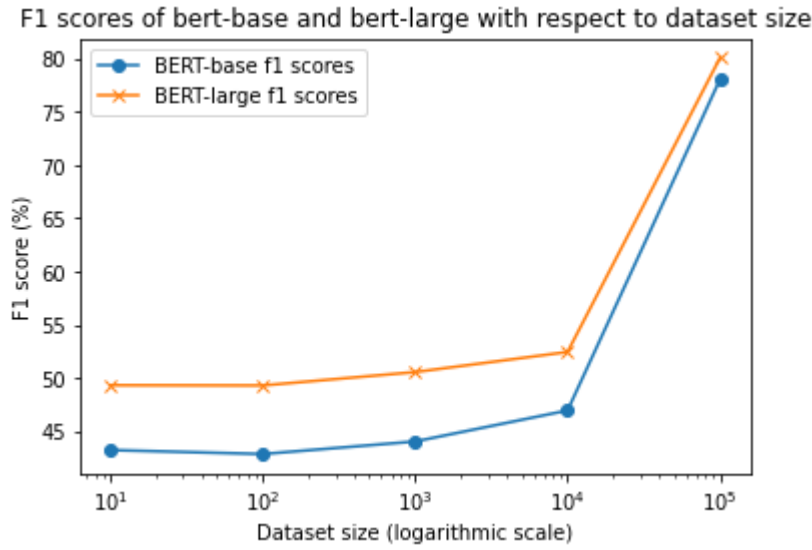


Figure 6-5: Performance of BERT Large and BERT Base versus training set size

Figure 6-5 summarizes the team's findings and justifications for choosing BERT Large over BERT Base for the task at hand. The data were generated by dividing the SQuAD 2.0 set into a validation set of size 50,000 and a training set of size 100,000. Each data point was computed by training the model on a subset of the training set with the corresponding size, and then measuring its accuracy in predicting the validation set via the F1 score metric for question answering (defined in the Glossary). For smaller dataset sizes, BERT Large tends to have higher accuracy, and after a sufficient training size, both models are roughly equivalent.

The BERT Large variant architecture has more parameters and is thus more expensive to train and operate but is also more accurate. Notably, it performs significantly better when the fine-tuning dataset is small. By contrast, the BERT Base variant architecture has fewer parameters and is thus less expensive to train and operate, but also less accurate. Hence, BERT Large was selected as the model architecture.

Table 6-3: BERT Large concept selection

Concept	Description	Ease of implementation (0–10)	Performance (0–10)	Resource efficiency (0–10)	Select (Y/N)
DeepPavlov library	Instead of fine-tuning BERT, use	10	8	0	N

DeepPavlov, which is already tuned					
Use BERT Large directly	BERT Large has more parameters and is more expensive to train and operate but more accurate	2	10	10	Y

The team had two choices for implementing the BERT Large deep learning model for question answering. The main deciding factors are given in Table 6-3. The DeepPavlov library provides a question answering method that abstracts away model details, which improves its ease of implementation score. Although this method is simpler to implement, it was discovered that installing and running the library itself uses significantly more disk space as well as memory compared to using BERT Large directly. This poses limitations on the memory usage of other necessary subsystems running on the same server. The direct use of BERT Large also results in better performance, as it allows introspection and tweaking of the inner workings of the model to provide slightly better performance for the team's use case.

### 6.2.2 Text Pre-Processing

Figure 6-6 gives an example of the basic use-case of the first step in text pre-processing which is to get rid of html tags and unusual characters, as well as to condense the text by removing repeated characters that provide no extra meaning. In this example, the html cross-reference tag is removed, and the sequence of newlines are converted to just a single newline.

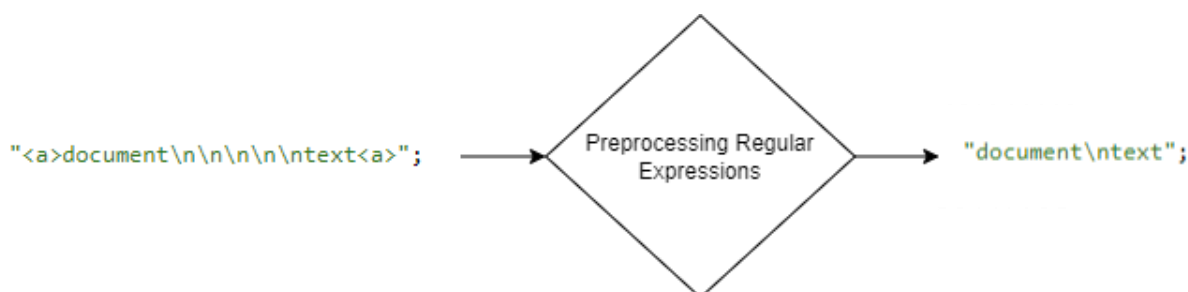


Figure 6-6: Preprocessing regular expression example<sup>1</sup>

Keywords are a good distinguishing factor between two separate documents. This design treats all verbs and nouns as keywords. The team utilized the Spacy keyword extraction library, described in Figure 6-7, in order to identify verbs and nouns in any given body of text. These can be utilized to establish a relevance score among a collection of documents and will therefore allow the ranking of documents by relevance.

<sup>1</sup> The “\n” back-slash n character sequence signifies to display the subsequent contents on the next line.

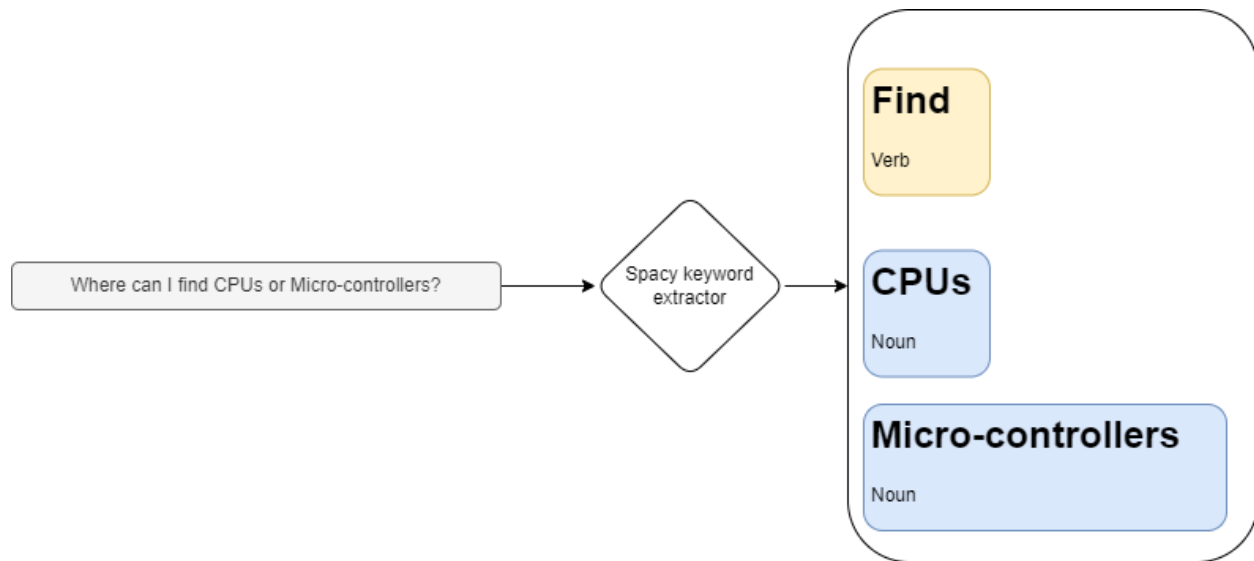


Figure 6-7: Named Entity Recognition using the Spacy Python library.

The last step in the text pre-processing pipeline is to convert the keywords into points in a vector space by the use of the word vectorization algorithm, Word2Vec [11]. This conjugate representation allows for a notion of similarity between the words that can be efficiently organized and queried through spatial indexing algorithms.

### 6.2.3 Inverted Index

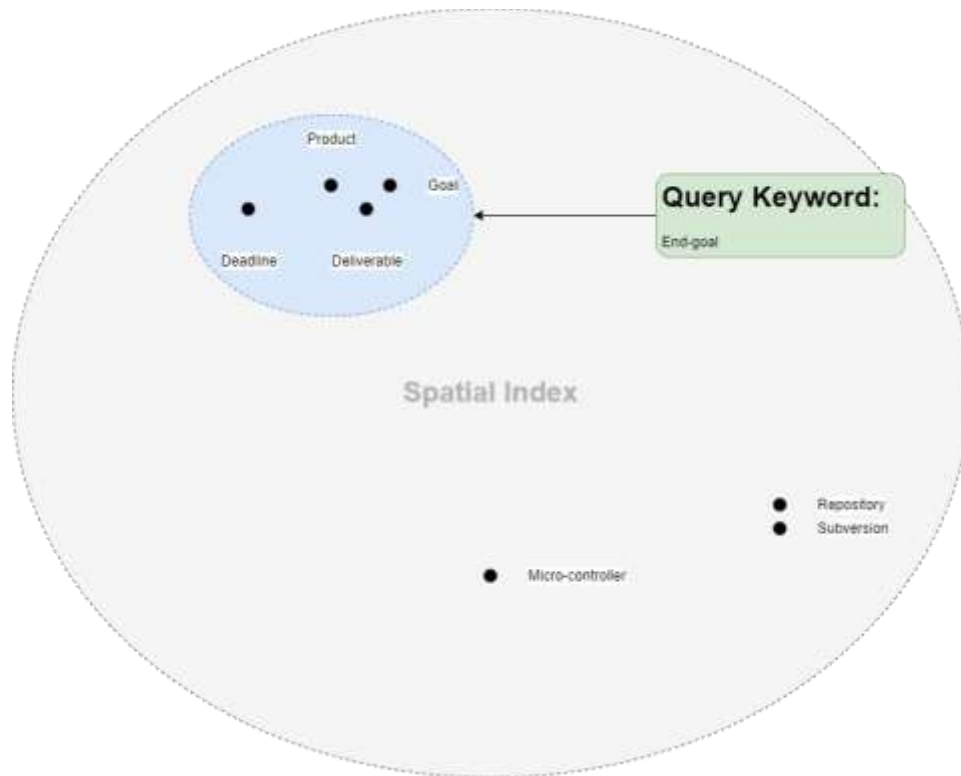
In the example shown in Figure 6-8, the set of documents is represented by the different texts enclosed in blue, and the mappings from keywords to documents are shown in the table on the right. This comes in handy for efficiently retrieving documents that contain a particular keyword. Its significance will be explained shortly.

Inverted Index		keyword	doc_id
1	This is SQL and Python and other fun teaching stuff	Python	1
		SQL	1
		This	1
		and	1
		fun	1
		is	1
		other	1
		stuff	1
		teaching	1
		More	2
2	More people should learn SQL from UMSI	SQL	2
		UMSI	2
		from	2
		learn	2
		people	2
		should	2
		Python	3
3	UMSI also teaches Python and also SQL	SQL	3
		UMSI	3
		also	3
		and	3
		teaches	3

Figure 6-8: Inverted index example



The question answering model itself has a relevance score for a document and its reliability in being able to draw a conclusion from it to answer the user's question. The problem with this approach is that it is infeasible to apply this question answering model to every document and then pick the best answer since, in practice, the set of documents is very large, and applying the model is a time-consuming operation. Estimating from the team's calculations, it could take up to a day to retrieve the answer using this approach. Therefore, it is necessary to first filter the set of documents based on a coarse relevance metric, to a significantly smaller set of documents, and then run the model on those. This is done so by establishing a relevance score by picking the closest documents to the keywords in the user's question. The spatial index, described next, is essential for implementing this method.



**Figure 6-9: Spatial Indexing demonstration**

In Figure 6-9, the vectors associated with each word are denoted as points. The position of the user specified keyword is calculated, and the closest keywords are retrieved. These keywords are then passed to the inverted index, shown earlier, to retrieve the documents that contain them. The team uses a lightweight experimental spatial indexing Python library called R-Tree to store and query the spatial index, as the algorithms involved in storing and querying large scale spatial data are out of the scope of this project. For simplicity, only a single keyword is shown here, but in a real scenario, there can be multiple keywords. To retrieve the N most relevant documents, the K nearest words for each keyword in the user's query are chosen and the documents for each of those words are collected into the set for each corresponding word. They are then ranked by their frequency among all of the keyword sets, and the N highest ranking documents are chosen. In case of ties, the lexicographically smallest document is prioritized. K and N are configurable constants. Larger constants will lead to a more extensive search and higher search times.

## 6.2.4 BERT Question & Answer Model

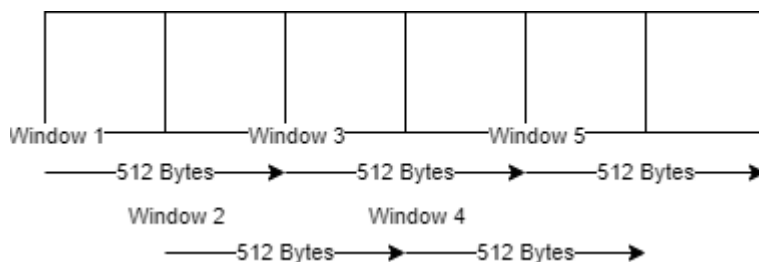


Figure 6-10: Sliding window algorithm illustration.

The BERT Q&A model can use at most 512 bytes of context data to answer a question. Many documents are larger than 512 bytes, so they cannot simply be fed into the model directly. Figure 6-10 illustrates the sliding window approach taken to apply the model to each document in 512-byte window increments and then pick the one with the highest confidence. The windows are partially overlapped in order to decrease the chance of important context being staggered across two windows. In the example, the document is 1.5 KiB in size and takes 5 windows to cover, meaning that the BERT model must be applied 5 times to complete evaluating this document.

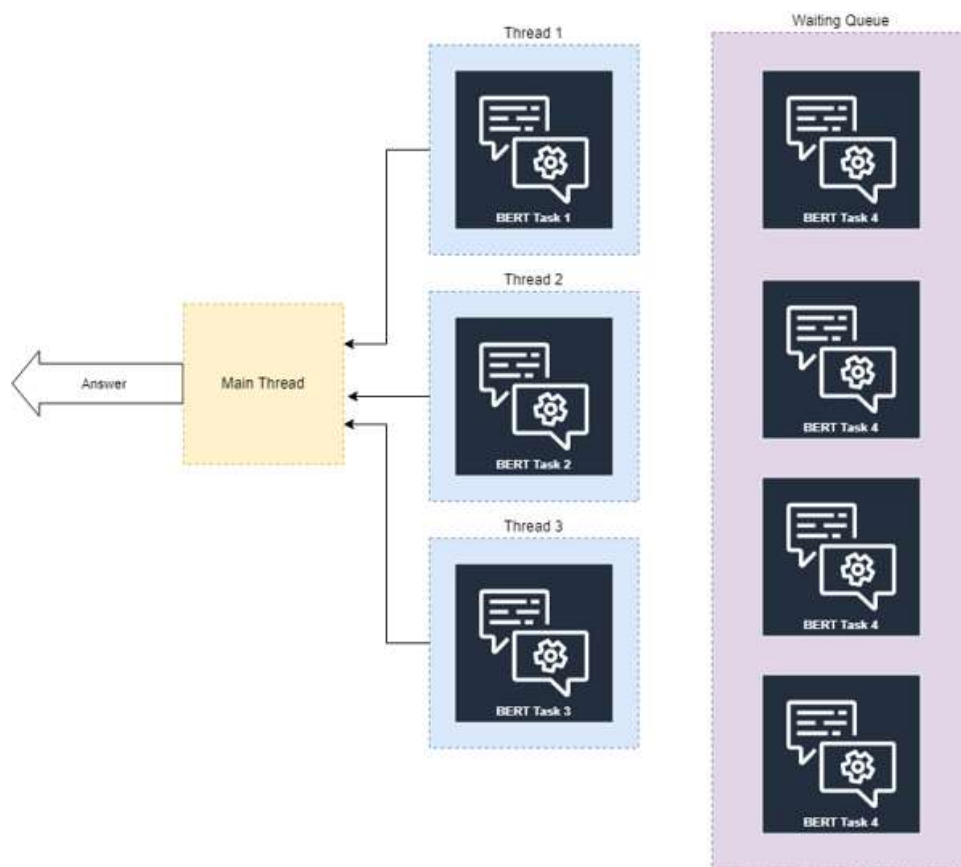


Figure 6-11: Application of thread-pool to parallelize application of BERT tasks.

Each application of BERT is considered as a BERT task in Figure 6-11. In order to improve the performance of the system with respect to time, the team has utilized a thread pool to parallelize BERT tasks. The example shows a scenario where the machine has 3 physical cores and a total of 7 BERT tasks. As a result, a thread pool of size 3 is configured to execute the series of tasks.



Because three tasks can be completed simultaneously, this configuration can theoretically reduce processing time to a third of the original time.

### 6.3 Rule-based chatbot

Rule-based chatbots take users along different paths in a tree of questions and answers based on their responses, with retrieval-based chatbots automatically gathering some of the information presented (for brevity, this report uses *rule-based* to refer to both). The team's goal was to adapt these concepts to make it as easy as possible for instructors and future teams to extend the rule system and also allow searching of information that does not fit well with answering a series of fixed prompts. For this, the team has prototyped a rule-based chatbot framework which includes a backend application that serves a conversational chatbot API based on an administration provided rule specification, as well as a graphical tool for constructing this rule specification. In this section, the subsystem design will be described in detail, and improvements suggested by the instructors will be discussed for guidance in future iterations of this project.

#### 6.3.1 System Design

The team's design to meet this need is a tree that mirrors the structure of the EDN wiki, each node representing a point in the wiki's hierarchy of pages (along with some associated keywords). The children of the tree's root are the broadest categories, such as course guidelines, purchasing procedures, and PE/CE information, that appear on the home page of the wiki. The user's initial query is sorted into one of these nodes by keyword similarity, or if no node is suitable, a message explaining this and suggesting Advanced Search instead is displayed.

Each node of the tree displays specified text, supplemented with a link to a wiki page if appropriate, to the user when reached. The text may contain an answer to the user's question or a prompt to ask for more information. This scheme is shown in Figure 6-12.

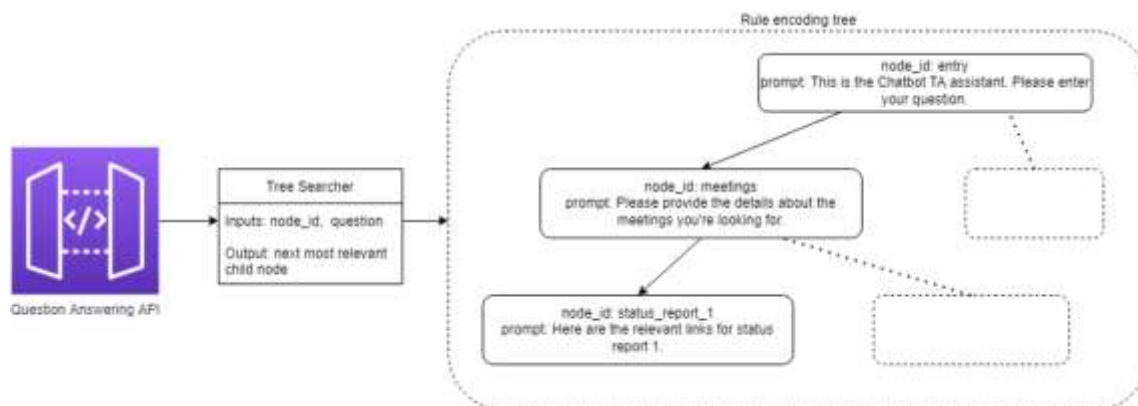


Figure 6-12: Simplified rule specification tree

However, for some information, keyword similarity matching is not enough. Keyword matching would help students find an assignment by name, but not find all assignments whose due dates are in the future, all assignments that are to be completed individually, or a combination. Given that four of the five most asked-for topics according to the Customer Needs Survey are related to assignments and in-class tasks, it makes sense to support these use cases specially.

Therefore, class periods and assignments are stored in tables separate from the main rule-based tree. This is a reasonable compromise that makes it equally efficient to search by name, date,

whether an assignment is graded, or anything else. Currently, the tables can only be searched via a Q&A API call separate from the one used for the main rule tree, and this is not supported by the frontend. Integrating these tables into the frontend is left for future teams.

The methods by which these rule- and retrieval-based systems are populated with wiki data is described in Sections 6.3.2 and 6.3.3.

### 6.3.2 Rule-Based Tree Construction

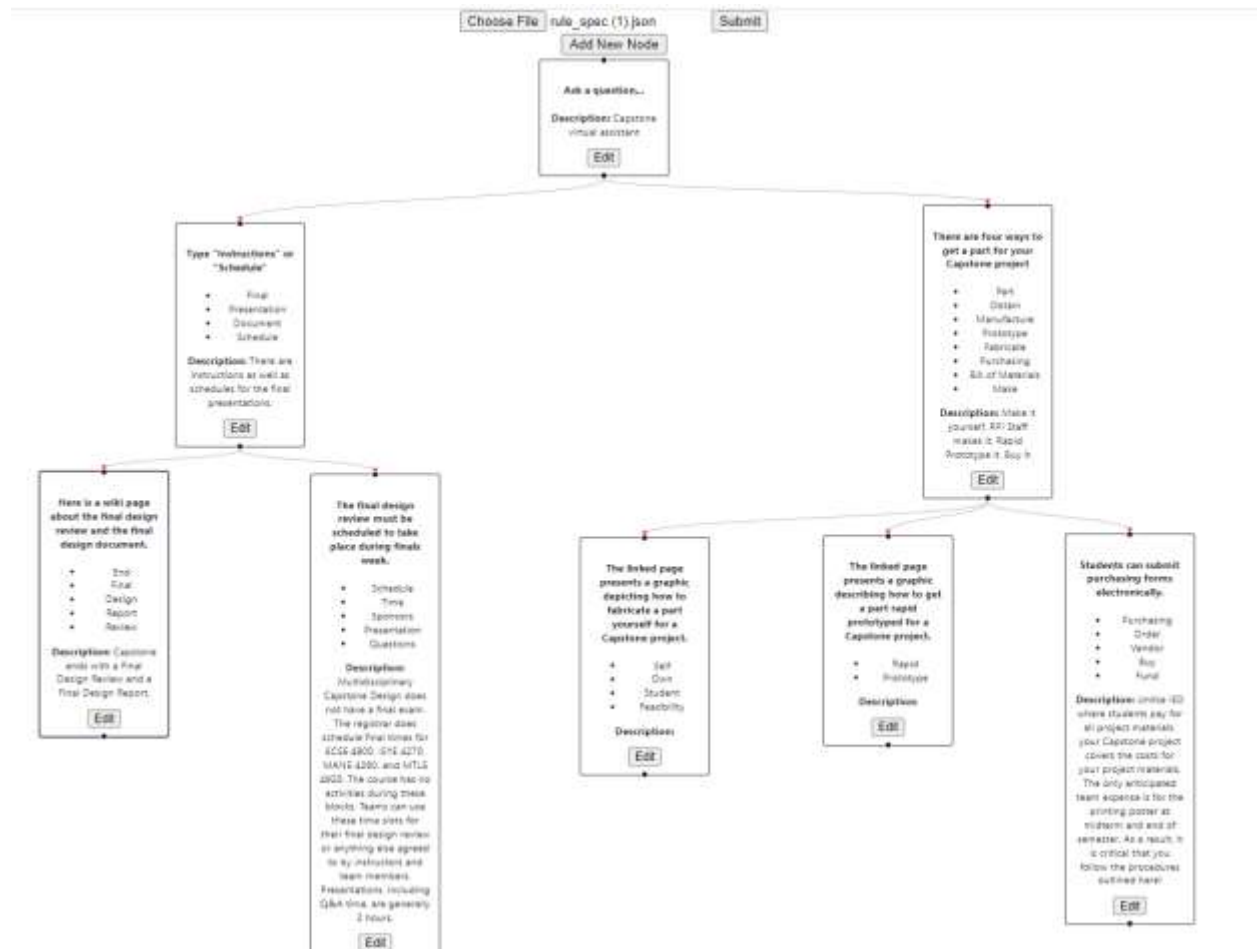


Figure 6-13: Rule specification tree example

Adding new nodes or edges to the tree is not inherently technical or programming-related. It is a task that course instructors could complete, given an intuitive tool. And since the rule tree should ideally be expanded as users find out important queries that Advanced Search doesn't handle well, the team found it worthwhile to write a Web app with a graphical user interface to facilitate this. The app can be used by both instructors and developers to continually improve the rule-based tree.

To begin, the user can start with either an empty tree or a tree specified by a JSON file. Then, the graphical interface allows the user to add or remove nodes and the edges between them. Each node must have a unique ID and keywords and can optionally include a prompt, description, and hyperlink. The keywords of a node are the words used with similarity matching to determine

whether to move to the node from one of its predecessors (the nodes with outgoing edges that point to this node). Afterward, the user can download their new tree as a JSON file.

While this system gives the user total freedom to restructure the rule tree, only developers with access to the chatbot's code can commit the JSON file containing the modified tree into the system. Until then, the modified tree has no effect on the chatbot itself.

### 6.3.3 Assignment and Class Table Construction

The assignment and class period tables are based on the data from the Tasks and Due Dates page [12] on the EDN wiki, which contains an HTML table with all the information. A Python script fetches the wiki page through the wiki's Redmine REST API and uses the BeautifulSoup library [13] to locate the headers, rows, and cells of the table. Based on a manually defined mapping of attributes (like name, description, or date) to column numbers, it knows which columns of the table represent which assignment or class attributes when constructing data structures to represent them. The script also scans for the cells with the background color that signifies graded assignments, as opposed to ungraded ones.

The script then caches the scraped data in a JSON file so that the Rule-Based Chatbot module can reconstruct it on startup without connecting to the Internet and fetching the Tasks and Due Dates page again. The module then exposes its table search and rule tree features through separate API functions.

This caching presents the issue of when to run the script, to keep up with updates to the Tasks and Due Dates wiki page. It would make sense to run it at the same time as the main scraping module, or even integrate it into that module, but these tasks are left for future semesters. For now, it must be run manually after every update to the wiki page.

### 6.3.4 Instructor Feedback on Rule Tree Editor Tool's Ease of Use

The goal of the rule tree editor tool is to provide a graphical interface to construct the knowledge base of Rule-Based Chatbot. The only user of this product would be the instructors of Capstone. For that reason, Dr. Kanai, the supervising Project Engineer for the team and a general maintainer of Capstone Design at RPI, was given the task of utilizing the graphical tool to load an existing specification file and add two additional nodes to it.

Dr. Kanai's full comments are attached in Appendix E: Instructor Comments on Rule Generation Tool. In summary, the instructions were not clear enough, so he did not fundamentally understand how the tree works to serve as the rule set of the chatbot. The rule tree is itself a very abstract idea, so describing it in words may or may not work well in imparting a fundamental understanding of its use. Instead, it would be best to include examples of segments of the tree and the chatbot traversing through that segment. The instructions on how to add, edit, and delete entities of the tree should also be accompanied by picture examples.

Lastly, Professor Kanai found that the process of constructing the tree allowed him to think of more ways to improve the wiki itself. For example, he found that new wiki pages needed to be added since he'd have to go to an external source to answer some of the questions he thought students might ask. This is good for Rule-Based Chatbot, as one of its main goals was to allow the instructors to improve the content within the EDN wiki over time.

## 6.4 Web Scraping

In order to fill the database with information from the EDN website, the chatbot uses a Web scraper to gather information (see Figure 6-14). The scraper interacts with the EDN website to navigate the site in order to extract the HTML code for processing. The HTML code is then parsed for the wiki page and forum content, which is then stored in a JSON file called “scraper\_output.json” for the BERT system to use. By setting up a cron job to run the tool daily, the chatbot is able to keep its databases up to date with the EDN website.

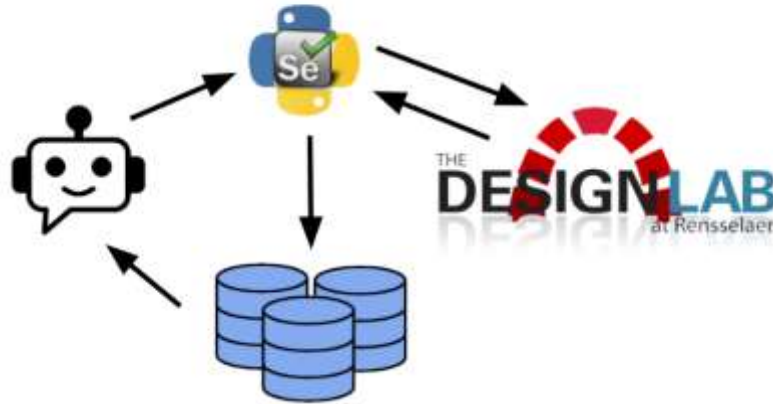


Figure 6-14: Web scraper flow chart

### 6.4.1 Concept Selection

There are several common libraries that can be used to implement Web scrapers, including Urllib, Requests, Scrapy, and Selenium. Ultimately, Selenium [14] was chosen for its browser automation features, which allows the program to interact with the EDN site and visualize the browser as the scraper progresses. Selenium’s ability to interact with the website is important because it allows the program to log into the site to check a specific project assigned to a user. This also allows it to scrape text that would normally be hidden until a user interacts with the page.

Visualizing the browser makes for an easier debugging process which is also very important, as this project will be a multi-semester endeavor with different people working on it with a varying skillset and familiarity with scraping. Selenium is also superior at scraping JavaScript. The selection process is depicted in Table 6-4.

Selenium also has the added benefit of built-in HTML parsing, whereas other scraping modules must parse HTML with the separate library BeautifulSoup [13].

Table 6-4: Scraper module concept selection

Concept	Description	Ease of Implementation (0-10)	Resource Efficiency (0-10)	Effectiveness (0-10)	Select
Requests	Basic scraping library	5	8	4	N
Selenium	Automatic browser	7	3	10	Y
Scrapy	Oriented for large projects	5	7	6	N

Urllib	Built in scraping library	3	8	6	N
--------	---------------------------	---	---	---	---

## 6.4.2 System Design

The EDN scraping command-line tool is a Python script that iterates through the EDN wiki and forums, scrapes text and tables from each page, and then stores the scraped information in a JSON file. It incorporates error logging using the Python logging module in order to store debugging information and program info, and the user has the option to customize the logging level and store the logging information to a file and/or print to the console. This allows the user to monitor the program as it runs and help with the debugging process.

The EDN scraper incorporates many command-line options for the purpose of specifying logging levels and outputs, specifying a specific project to scrape, specifying whether the forums of a project should be scraped, and showing the browser as the program runs. The scraper uses a Python module called Argparse [15] that enables a program to easily implement command-line arguments. This allows the scraper to be run either by a user in the command-line, or as part of a shell script. This customizability was implemented in order for an instructor to be able to quickly update the chatbot databases for a specific project on EDN, as opposed to having the program scrape the entire website unnecessarily.

The scraping tool is able to scrape the wiki and forums of a specific project when the user specifies the project, their username, and their password in the command line. The program finds the link to the project by logging into EDN using the user's username and password, going to the user's project page, and sorting through the attached projects until it finds the project corresponding to the one specified in the command line. The program is also able to visualize the browser as the program runs, which makes the debugging process for future students working on the chatbot easier by allowing them to watch the browser instead of adding print statements to the code and reading through the log.

## 6.4.3 Wiki Scraping

EDN wiki page information is mostly presented in the form of text and tables encoded in HTML (see Figure 6-15). Text content is mostly implemented using paragraphs `<p>` and list items `<li>`, so the scraper searches for those HTML objects specifically to scrape and store. The program also stores all tables associated with the page.

```
* <h1></h1>
* <p>
  "Capstone ends with a Final Design Review and Final Design Report which are the CULMINATION of your semester project. At this stage, the
  team should have an in-depth knowledge and understanding of the problem and your proposed design solution. All of the prior steps have lead
  to this point where your team will present the final report, which includes both a written technical report and oral presentation."
</p>
* <p>
  "The written report and final presentation should be prepared in cooperation with all members of the project team. Each team member will be
  evaluated on their content contributions to the report and design review. In this regard, the team should prepare a summary work breakdown
  structure for both the report and presentation that clearly indicates individual contributions."
</p>
* <ul>
  * <li>
    :marker
    "Schedule for this semester's reviews is posted here - "
    <a class="wiki-page" href="https://edn.fitch.edu/teams-support-des/vtds/final-design-review">Final Design Review</a>
    </li>
  * <li>
    :marker
    <strong>Grading rubric</strong>
    " available on the "
    <a class="wiki-page" href="https://projects.fitch.edu/teams-support-des/vtds/tasks-and-due-dates">Tasks and Due Dates</a>
    " page."
    </li>
</ul>
* <div>
  <a href="https://edn.fitch.edu/teams-support-des/vtds/final-design-review">Final Design Review</a>
</div>
* <div>
  "Final Design Review"
  <a href="https://edn.fitch.edu/teams-support-des/vtds/final-design-review">Final Design Review</a>
  <a href="https://edn.fitch.edu/teams-support-des/vtds/final-design-review">Final Design Review</a>
</div>
```

Figure 6-15: HTML breakdown of an EDN wiki page

Due to the lack of an easily accessible list of every wiki and forum page on the EDN site, it was determined that the best way to implement the scraping tool in order to capture the most information was using a recursive approach. In order to keep runtime down and prevent the program from scraping the same page twice, a list of previously scraped pages is kept. The script searches the page for other EDN wiki page links and checks it against the list of previously scraped pages. If the link is not contained within that list, the function calls itself on the link to scrape that page.

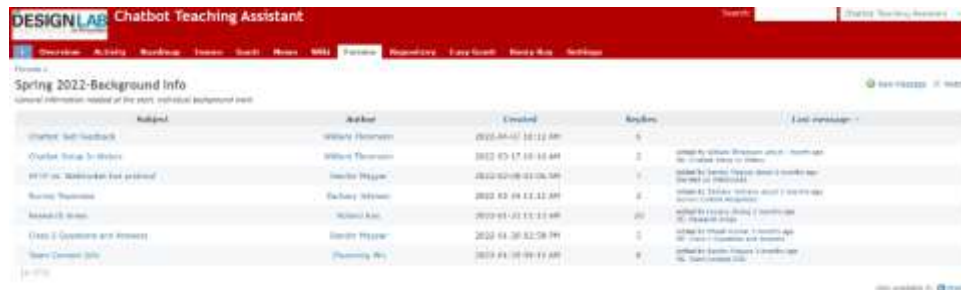
#### 6.4.4 Forum Scraping

The home page of the EDN forum (Figure 6-16) is structured as a table with a column for each forum and a brief description, a column keeping track of the number of topics per forum, another column with the number of messages in the forum, and a last column detailing the date and author of last message written in that forum. Upon clicking a link in the first column, the second page (Figure 6-17) is often (though not always) another similarly structured table with columns detailing the subject, author, date created, number of replies, and last message in the thread. Clicking a link in the subject column will bring the user to the associated thread (Figure 6-18).



Topics	Topics	Messages	Last Message
Spring 2022 VICE Blog A place for your thoughts on our overall, project feedback and announcements.	7	8	Added by: AutoPostBot 24 hours ago Added by: VICE 24 hours ago
Spring 2022 Background Info General information related to the start, individual background info.	7	11	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022 Project Management Information used to manage the project such as development of meetings, meeting minutes, etc.	15	108	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022 Design For planning and sharing technical design-related information.	4	45	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022 Development of Tools A place to collaborate on the development of the VICE tool to plan and develop software when the need arises.	0	0	
Spring 2022 Mission A place to collaborate on the mission requirements and to plan and develop software when the need arises.	1	1	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022 Project Management Information used to manage the project such as development of meetings, meeting minutes, etc.	1	1	Added by: VICE 24 hours ago Added by: VICE 24 hours ago

Figure 6-16: EDN forums initial page



Subject	Author	Created	Replies	Last Message
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago
Spring 2022-Background Info	William Thompson	2022-04-01 10:12 AM	5	Added by: VICE 24 hours ago Added by: VICE 24 hours ago

Figure 6-17: EDN forums second page table





Figure 6-18: EDN forums thread

Due to the way the forums are structured, the clicking the first column of links will eventually bring the user to the forum thread. Keeping this in mind, the program's function associated with scraping the forum first attempts to scrape a table. If it is successful, it saves this data and then iterates through each link in the first column by calling itself on the link. If it is not successful at finding a table, the program then knows that this page contains a thread and scrapes the page for text.

### 6.4.5 User Manual

There are nine command-line arguments that can be specified. Specifying no command-line arguments will result in a warning to the user that reminds them that they have not chosen any arguments. Command line arguments do not need to be placed in any specific order. To see a complete list of command line options in the terminal, the command can be run with `-h`.

In order to specify a specific project to be scraped, `-r` or `--project` should be added to the command line, followed by the name of the project. The project should be spelled exactly the way it's spelled on the user's project page, as that's what the program is looking for. In order for this option to work, the browser needs to be logged in to the user's EDN account, so a username and password need to be added to the list of command-line arguments. The user's username can be specified by adding `-u` or `--username` followed by the username to the command line, and the user's password can be specified by adding `-x` or `--password` followed by the password. In order to have the tool scrape the project's forums, `-m` or `--forums` can be added to the list of arguments.

The capstone support wiki will be scraped by the program by default, though this can be prevented by using option `-k` or `--skip`.

Information logging can be set to one of five levels, with each subsequent level including more information. For example, setting the level to 50, or critical, will track all information sent to the critical level and nothing else. However, setting the level to 10, or debug, will track all information on every higher level. The five levels in order of least to greatest are debug, info, warning, error, and critical, with the associated number starting at 10 and incrementing by 10 until level 50. To tell the tool which level the logger should be set to, `-l` or `--level` should be added to the command line, followed by the level. The level can be specified in all lowercase, all uppercase, the first letter in upper or lowercase, or by the associated number. For example, to set the level to warning, anyone of the following can be used: warning, WARNING, w, W, or 30. The generated log can be printed to the terminal using `-p` or `--print`, and it can also be saved to a log file by adding `-f` or `--file` to the command-line, followed by the name of the log file to be saved to.

The last command-line argument that can be specified is `-s` or `--show`, which turns the Web driver's headless mode off, visualizing the chrome browser as the program runs. This option does not work on the chatbot server, so future students working on the chatbot should download the code and run it on their own computers in order to utilize this option.

## 6.5 Data Analysis and Visualization

The team uses Python's PANDAS (Python Data Analysis) library [16] to load data from chatbot logs and Matplotlib's [17] histogram feature to visualize the frequency of search entry keywords on a plot of time. The analysis is done on the user survey results to demonstrate the concept of feedback provided to instructors and students. Figure 6-19 shows the results.

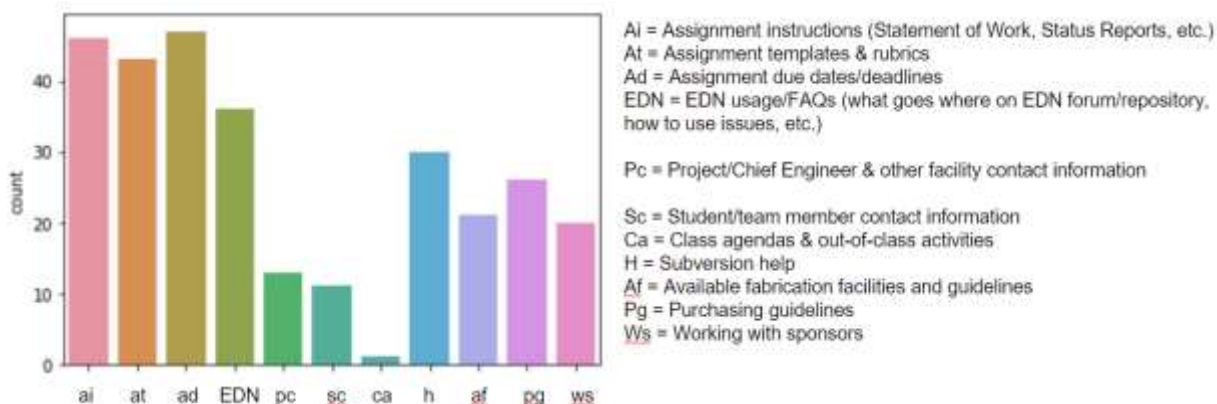


Figure 6-19: Visualization with student answers

Another part of the program is applied to randomized data to showcase the frequency plot format (Figure 6-20).

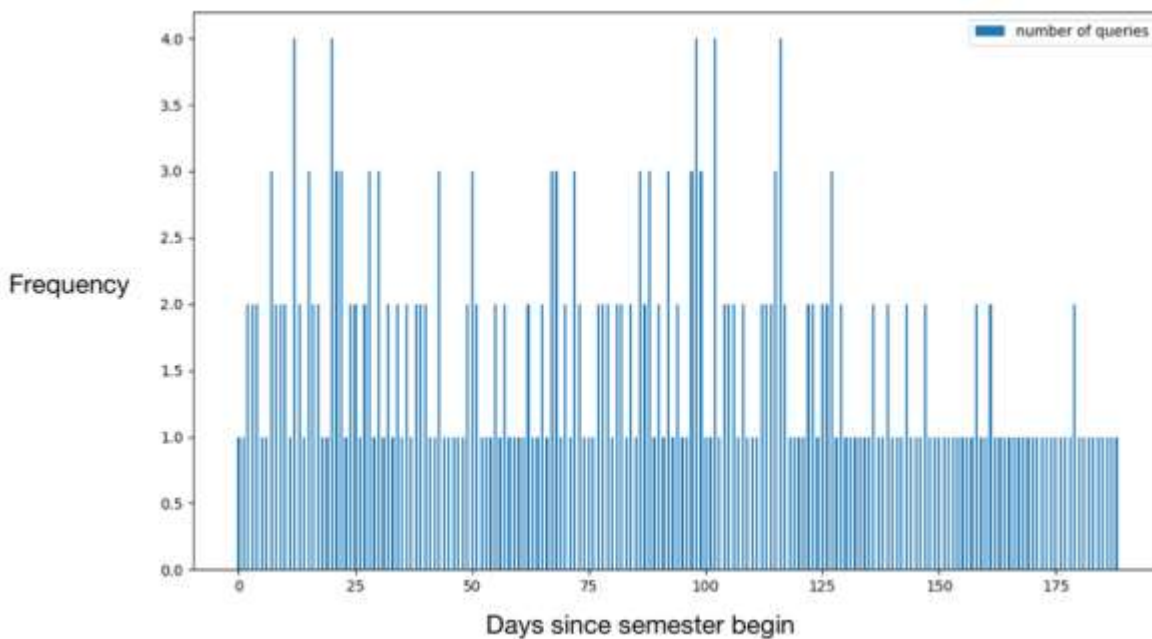


Figure 6-20: Frequency plot format with random data



## 6.6 Database Logging and Categorization

This project will take multiple semesters, so the flexible database MySQL was used to store data in a format easy for future teams to work with. MySQL contains multiple engines; each has its pros and cons. For each different table, MySQL can apply different engines based on the user or developer's needs. For now, the chatbot will not constantly transact between queries, which requires locking and unlocking to resolve concurrency. Thus, the MyISAM engine will store all the answered and unanswered questions, since this engine stores data with fixed length to give better performance on reading.

## 7 System Evaluation

Four tests were created to test the ability of the chatbot: question answering accuracy, user friendliness, system robustness, and user interface. Each test was set up in a similar fashion with user prompts being decided beforehand for the user to input.

The first test, question answering accuracy, compared the output from the chatbot to the predicted sentence obtained from reading the EDN page. An F1 accuracy test was used to evaluate the outcomes. The results from this test are shown in Table 7-1. The average F1 accuracy score of ~0.81 indicates this objective of answering questions to a satisfactory level has been met.

The second test, user friendliness, evaluated the time to answer each previously selected prompt. The results of the user friendliness test are shown in Table 7-2. With an average result of ~5.8 seconds to resolve each inquiry, the chatbot proved to be an effective tool to lower the time it takes a student to find answers about Capstone Design. A secondary user friendliness test was done comparing the time to answer each inquiry between the chatbot and the EDN search tool. The results of this test, shown in Table 7-3, led to the same conclusion.

The third test created was to evaluate system robustness. This test has multiple users input inquiries at the same time and tests the ability for the chatbot to still perform to an adequate standard. Due to the time constraints of the project, this test was not performed.

The final test was an overall feedback evaluation from the user about the user interface. Test users were presented with a survey after completing the previous tests. This survey was designed to garner specific feedback about the user interface and asks to provide recommendations for improvements going forward. The users were asked whether the interaction was satisfactory (yes or no). Then, they were asked to select all aspects of the experience they thought should be improved. This question provides useful feedback to future teams who are evaluating the chatbot on their own and looking for areas to improve. The final question also provides an opportunity for the user to directly interact with the current and future design teams on what specific improvements they would like to see. Due to time sensitivity, this test was unable to be conducted.

Future steps and improvements will be clearer when the final two tests are performed, and it is a recommendation that finishing the tests are the first steps done if a new team takes on this project. Completing these final tests will give insight into which subsystems need improvement. Recommendations for improving the search tool for in class activities and assignments, along with obtaining a faster server, and continuing to build out the rule-based tree are recommended along with the feedback gathered from completing these tests.

Table 7-1: Question answering accuracy

Test No.	Question	F1 accuracy	Predicted Answer	Correct Answer
0	How do I order parts?	0.173913043	actuating components like motors and relays that exceed the current or voltage capabilities	['Purchasing']
1	How do I order parts?	1.076923077	fabricating things yourself or having our shop do the fabrication	['fabricating things yourself or having our shop do the fabrication']
2	What are vendors of microcontrollers ?	0.125	CABINET Microprocessors	['Raspberry Pi\nArduino Mega, Uno, Due, etc\nATtiny\nLITEC']
3	What is the final deliverable in capstone?	0.916666667	Rapid Prototyping Resources Process for creating parts for Ca	['Rapid Prototyping Resources\nProcess for creating parts']
4	What is the final deliverable in capstone?	1	hard copy drawings	['hard copy drawings']
5	Where are the templates for design reports located?	1.125	Tasks and Due Dates page	['Tasks and Due Dates page']
6	Where are the templates for design reports located?	0.285714286	on a separate page	['the attached PowerPoint file']
7	Where are the templates for design reports located?	1	PowerPoint	['PowerPoint']
8	Where are the templates for	1	CAD folder	['CAD folder']

	design reports located?			
9	Where can I find microcontrollers or CPUs?	1	online	['online']
10	Where can I find microcontrollers or CPUs?	1	P12 file Register devices	['P12 file\nRegister devices']
11	Where can I find microcontrollers or CPUs?	1	+ ESP8266 combination	['+ ESP8266 combination']
		<b>Overall F1 Accuracy Score</b>		
		<b>0.808601423</b>		

**Table 7-2: Advanced Search response times**

Test No.	Question	Time (seconds)
0	What are vendors of microcontrollers?	6.634665
1	Where can I find microcontrollers or CPUs?	8.226435
2	What is SVN?	0.047296
3	What is the final deliverable in capstone?	5.289309
4	Where are the templates for design reports located?	8.583716
5	How do I order parts?	7.496868
6	How do I resolve an issue?	5.698333
7	How do I upload to the repository?	4.16966
8	What are the learning goals of capstone?	6.379
		<b>Average Time (seconds)</b>
		<b>5.836142444</b>

**Table 7-3: Total time to get an answer using Chatbot TA vs. existing EDN search**

Student query	EDN Search (s)	Chatbot (s)	Difference (s)
Where are the templates for design reports located?	30	20	-10
Where is the Statement of Work?	15	38	+23
Where do I submit the Statement of Work?	65	10	-55
Where do I submit the Preliminary Design Review, PDR?	151	22	-129
When is Phase 2 due?	18	20	+2

Where are the guidelines for purchasing located?	20	8	-12
What is the SMART acronym?	30	12	-18
Which Vendor has microcontrollers?	233	28	-205
How do I resolve an issue?	81	45	-36
How do I upload to the repository?	160	25	-135
How do I schedule a sponsor meeting?	81	50	-31

## 8 Significant Accomplishments and Open Issues

The team fulfilled all objectives that were planned at the beginning of the semester. These include the in-scope deliverables that were specified in Section 2.2. The team was also able to make progress on the first listed out-of-scope deliverable, which was to be able to answer iterated queries, or in other words, a conversational chat bot rather than having each question be completely independent. This was done using the Rule-Based Chatbot subsystem, discussed in Section 6.3.

Although the team had a thorough test plan, described in Appendix C: System Evaluation Plan, only some preliminary alpha tests were completed using teammates as subjects. In future iterations, there should be beta tests where the subjects are students from Capstone Design who were not involved in the project's development. Additionally, as discussed in the results from Dr. Kanai in Section 6.3.4, the instructions for the rule construction tool need to clearly describe the way by which the rules are transformed into a chatbot.

The testing results show that the advanced search API takes several seconds to return a response. Because it is doing CPU-intensive computations, it will be extremely slow when many users try to use it simultaneously. For that reason, a future improvement would be to distribute the API such that several machines are running the server code and can handle user requests.

## 9 Conclusions and Recommendations

Preliminary results show that the Chatbot Teaching Assistant is already a significant improvement over the alternatives. The Advanced Search feature provides breadth of coverage, answering queries using the full Capstone Support Wiki, and is more tolerant of synonyms and query wording than the search built into the wiki. The Rule-Based Chatbot provides finer control and more reliability for a subset of these queries, as well as a clear direction for improvement. As the chatbot is used more, instructors and developers can add more paths to the rule-based tree to continually improve its performance on the most important queries.

The team met its semester objectives, but much remains to be done. Future teams should continue expanding the rule-based tree, integrate the assignment and class period search feature described in Section 6.3.3, find a better hosting setup for the bot, and finish carrying out the team's system evaluation plan to identify other areas for improvement.

According to Section 8, the system should be distributed to multiple servers to keep Advanced Search fast while there are multiple simultaneous users. At first, the team hosted it on an RPI virtual machine, but to get more memory, the team temporarily switched to a commercial hosting service. In order to continue this project in future semesters, a permanent host must be found: either wait for a more capable RPI virtual machine or find a way to fund continued commercial hosting. The latter is recommended if the system becomes widely used, as it seems more

amenable to distribution across multiple machines, based on the shortage of RPI virtual machines this semester.

In conclusion, the Chatbot Teaching Assistant has the potential to be a truly useful tool for students and instructors of Capstone Design. It should be a strong candidate for continued development by future teams.

## 10 References

- [1] D. Zhang, S. Mishra, E. Brynjolfsson, J. Etchemendy, D. Ganguli, B. Grosz, T. Lyons, J. Manyika, J. C. Niebles, M. Sellitto, Y. Shoham, J. Clark and R. Perrault, "The AI Index 2021 Annual Report," arXiv Preprint, 2021.
- [2] A. Banney, "You can now file your tax return with help from a virtual chatbot," 18 June 2018. [Online]. Available: <https://www.finder.com.au/you-can-now-file-your-tax-return-with-help-from-a-virtual-chatbot>. [Accessed 2022].
- [3] N. Joshi, "Choosing Between Rule-Based Bots And AI Bots," 23 February 2020. [Online]. Available: <https://www.forbes.com/sites/cognitiveworld/2020/02/23/choosing-between-rule-based-bots-and-ai-bots/>. [Accessed 2022].
- [4] E. Adamopoulou and L. Moussiades, "An Overview of Chatbot Technology," in *IFIP Advances in Information and Communication Technology*, Cham, 2020.
- [5] H. T. Hien, P.-N. Cuong, L. N. H. Nam, H. L. T. K. Nhung and L. D. Thang, "Intelligent Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, Danang, 2018.
- [6] T. Berners-Lee, "The Original HTTP as defined in 1991," 1991. [Online]. Available: <https://www.w3.org/Protocols/HTTP/AsImplemented.html>. [Accessed 10 February 2022].
- [7] J. Benyovski, "Using Websockets with the Webex JavaScript SDK," Cisco Systems Inc., 26 September 2019. [Online]. Available: <https://developer.webex.com/blog/using-websockets-with-the-webex-javascript-sdk>. [Accessed 10 February 2022].
- [8] M. Schmitz, "[How To] Building a Simple Webex Chatbot with Python Websockets & OpenWeather APIs," 18 November 2021. [Online]. Available: <https://0x2142.com/how-to-building-a-basic-webex-chatbot/>. [Accessed 10 February 2022].
- [9] RPI Chatbot Teaching Assistant team, "Needs and Requirements Workbook," Spring 2022. [Online]. Available: <https://designlab.eng.rpi.edu/svn2/dl-chatbot-ta/working/Background%20Information/Needs%20and%20Requirements%20workbook.xlsx?p=47>.
- [10] RPI Chatbot Teaching Assistant team, "User Survey," Spring 2022. [Online]. Available: [https://docs.google.com/forms/d/e/1FAIpQLSeFOB\\_Bw5VOvvijO0ZUIzJN51iWaAFOEDsGdVtEjC5vYSFA7Q/viewform](https://docs.google.com/forms/d/e/1FAIpQLSeFOB_Bw5VOvvijO0ZUIzJN51iWaAFOEDsGdVtEjC5vYSFA7Q/viewform).
- [11] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *International Conference on Learning Representations*, 2013.
- [12] Capstone Support team, "Tasks and Due Dates," [Online]. Available: [https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/Tasks\\_and\\_Due\\_Dates](https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/Tasks_and_Due_Dates).
- [13] L. Richardson, "Beautiful Soup Documentation," [Online]. Available: <https://beautiful-soup-4.readthedocs.io/>. [Accessed April 2022].
- [14] J. Huggins, "Selenium," [Online]. Available: <https://www.selenium.dev/>. [Accessed April 2022].

- [15] Python Software Foundation, "argparse," [Online]. Available: <https://docs.python.org/3/library/argparse.html>. [Accessed April 2022].
- [16] W. McKinney, "pandas," [Online]. Available: <https://pandas.pydata.org/>. [Accessed April 2022].
- [17] J. D. Hunter and M. Droettboom, "Matplotlib," [Online]. Available: <https://matplotlib.org/>. [Accessed April 2022].

## 11 Appendix A: Initial Deliverables and Dates

Table 11-1: Initial deliverables and dates

Deliverable	Planned date
Working trivial chatbot running on a server that can join the space when pinged with basic requests	2/7/22
EDN assistant functionality accessible through chat feature.	4/11/22
Provide means for instructor to update chatbot database	4/11/22
Bot can answer one question one answer questions	4/11/22
Stretch goal: bot can prompt the user for more information to answer more complex queries	



## 12 Appendix B: Project Plan

The project can be split into four main components: frontend, backend, implementation, and testing. These components will be carried out in order, with people not assigned to one group working on the next. Finishing all frontend work must be completed before implementation, along with a majority of features being completed. Testing can only be done once implementation has finished. Each task is critical to be completed to achieve the team deliverables. The order of tasks in each subcomponent is as they are listed. Each team member is assigned approximately five tasks.

Frontend: Will, Zachary, Roland, Sandor, Megan

Tasks include:

- Survey (will create customer needs for chatbot)
- Determine customer needs from survey (make sure chatbot is useful)
- Brainstorm new use cases for chatbot (addition features that could be added)
- Provision virtual machine (set up VM so members can ssh to deploy code)
- Upgrade host server Web-based UI (Need enough RAM to run chatbot)
- Determine model for data base (data must be stored and used for feedback)
- Set up chatbot UI through Webex (Webex will host the bot)
- Come up with one other developed ML backend (important to have back up option)
- Test frontend of Webex bot (ensure basic bot is running before adding features)

Backend: Mitesh, Sandor, Jonah, Roland, Megan, Leyang, Zhuoming

Tasks include:

- Parse keywords (search tool)
- Syntax analysis tree with convolution (chatbot path)
- Define Process for Unanswerable Questions (need path if question cannot be answered)
- Conduct F1 Score Tests on Sample Question Set (compare CNN to BERT)
- Build algorithm based on word vector (search tool)
- Machine learning algorithm that groups queries of unanswered questions by similarity (more complicated questions)
- Get machine learning Q&A algorithm up and running on an http endpoint (non-hard coded questions)
- Implement raw text page ranking (search tool)
- Reformat scraped data to make it more usable for ML (search tool)
- Create query for unanswered questions (need a response to all inputs)
- SQLite process (store questions)
- Create scraper (search tool)
- Chron Job set up (Update system)
- Create rudimentary database (store answered and unanswered questions)
- Create Instructor Dashboard (way to view the unanswered questions)
- Develop NLP tools to process data entries (Update system)
- Have EDN scraper search specified projects (search tool)
- Find out how to map user input keyword to EDN resource keyword (search tool)

- Write Documentation for NLP (update system)
- Define a review process for end users to evaluate performance (way to tell if chatbot was helpful)
- SQLite Tile to add admin answers to each group of unanswered questions (feedback for unanswered questions)

Implementation: Jonah, Mitesh, Roland, Megan, Sandor

Tasks include:

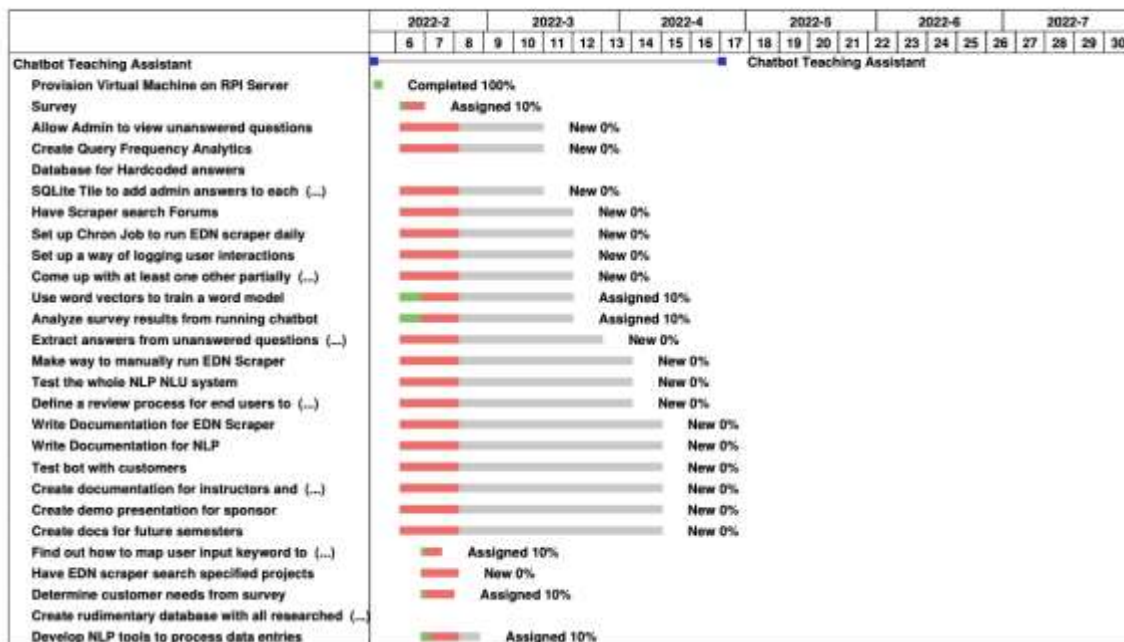
- Plug the whole system into the chatbot (all features must be added to Webex)
- Implement Query Similarity Interface (interface needs to be added to Webex)
- Online Setup (set up online database to work with chatbot)
- Collaboration with Front end on implementing it to Webex (work to get chatbot running)
- Deploy Instructor Facing Interface Plan (for unanswered questions and feedback)

Testing: Will, Zhuoming, Leyang

Tasks include:

- Create Report on Collected Data and User Guidance (feedback for future teams to examine)
- Test with small sample within team (see if deliverables can be met)
- Build Algorithm to categorize data (build analytics tools to analyze performance)
- Create feedback form for users who initially test (insight into how useful bot is)
- Test bot with customers (full test to see functionality)
- Create docs for future semesters (document further steps/struggles for future teams)
- Create demo presentation for sponsor (part of assignment)

#### Chatbot Teaching Assistant



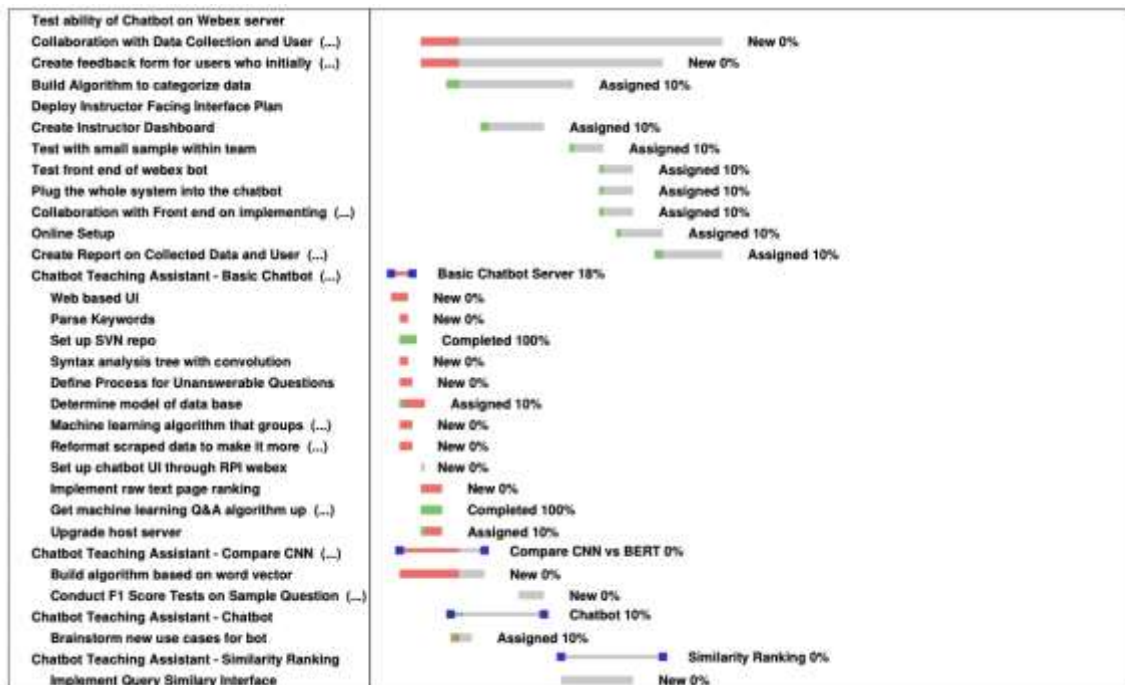


Figure 12-1: Gantt chart

## 13 Appendix C: System Evaluation Plan

### 13.1 User friendliness

**Table 13-1: Average time for students to arrive at a useful answer using the chatbot. Common student queries are from surveys.**

<b>Student query</b>	<b>1 (s)</b>	<b>2</b>	<b>Average (s)</b>
Where are the templates for design reports located?	20		
Where is the Statement of Work?	38		
Where do I submit the Statement of Work?	10		
Where do I submit the Preliminary Design Review, PDR?	22		
When is Phase 2 due?	20		
Where are the guidelines for purchasing located?	8		
What is the SMART acronym?	12		
Which Vendor has microcontrollers?	28		
How do I resolve an issue?	45		
How do I upload to the repository?	25		
How do I schedule a sponsor meeting?	50		

**Table 13-2: Average time for students to arrive at a useful answer by searching the wiki manually. Common student queries are from surveys.**

<b>Student Query</b>	<b>1 (s)</b>	<b>2</b>	<b>Average (s)</b>
Where are the templates for design reports located?	30		
Where is the Statement of Work?	15		
Where do I submit the Statement of Work?	65		
Where do I submit the Preliminary Design Review, PDR?	151		
When is Phase 2 due?	18		
Where are the guidelines for purchasing located?	20		
What is the SMART acronym?	30		
Which Vendor has microcontrollers?	233		
How do I resolve an issue?	81		
How do I upload to the repository?	160		
How do I schedule a sponsor meeting?	81		

Table 13-1 and Table 13-2 summarize the team’s strategy to evaluate the user friendliness of the student interface. Since the goal is to improve navigation of the wiki, the tables will be used to estimate the speed-up in getting an answer resulting from the chatbot for common questions. Each corresponding numbered column should ideally be tested by the same student, and each column should be obtained from a different student to reduce biases from any one student.

### 13.2 Question answering accuracy

Accuracy will be evaluated through measuring two types of F1 scores, shown in Equation 13-1 and Equation 13-2. As explained in the Glossary, these are commonly used to measure the accuracy of classification and question answering models, respectively.

$$F1 = 100 \times \frac{TP}{TP + \frac{FP + FN}{2}}$$

Equation 13-1: F1 score for answerability classification accuracy

$$F1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

Equation 13-2: F1 score for question answering accuracy

The team will compile a list of questions, context passages, and corresponding segments of the context passage that would be an acceptable answer, or no acceptable answers if the question is not answerable using the context passages. The process of aggregating multiple tests into a final F1 classification and F1 question answering accuracy score is demonstrated in Table 13-3. The formulas above, used for calculating the two different types of F1 scores, are described in more detail in the Glossary.

Table 13-3: Summary of question answering accuracy evaluation procedure

Test Number	Question	Context Passage	Answers	TP	FP	FN	Recall	Precision
...	...	...	...	...	...	...	...	...
							<b>F1 – Classification Score</b>	<b>F1 – Question Answering Accuracy Score</b>
							(Compute F1 score using total tallies of <b>TP</b> , <b>FP</b> , and <b>FN</b> )	(Average of F1 scores of each test)

### **13.3 Resource efficiency**

The team will evaluate how efficiently the system utilizes computational resources by measuring the time taken to retrieve an arbitrary result from the chatbot. For 10 trials, a random query from Table 13-1 will be chosen and an HTTP request to the question answering API will be timed using the curl UNIX tool. The team chose to run 10 trials because this should be enough for the latency behaviors of all the questions to be exhibited. This testing can be automated using the subprocess module in Python, which gives shell access. The average time required to complete the request will be recorded as the overall system latency which includes the time the question answering subsystem takes to generate an answer, as well as the travel time over the network.

### **13.4 Load handling**

In the Current Semester Objectives (Section 2.2), the maximum simultaneous users to handle was set at 100. The team will utilize multi-processing in Python to send 100 randomized questions in parallel to the question answering API. This can be accomplished with the Python subprocess library which enables the testing team to spawn multiple independent UNIX processes which can then send a random question to the API and measure the time taken to get a response. The maximum latency as well as the standard deviation will be recorded. A high outlier, standard deviation, or a failure to get a response on a process would indicate problems with load handling.

## 14 Appendix D: Ethical and Professional Responsibilities

Issues	Impact (1, low–5, high)	Description of impact and related project decisions
Public Health, Safety, and Welfare	NA	This issue doesn't apply to our project as the chatbot is incapable of impacting the wider public health, safety and welfare. It is limited to RPI community and its members
Global	4	The chatbot is accessible to all students and faculty providing they are on RPI VPN. This allows international students and other member of the RPI community to utilize the chatbot even though they are not physically on campus
Cultural	5	During our semester we found the EDN and information it provided was not straightforward and sometimes confusing. Having a chatbot that can answer common questions that students have, while also providing insights to instructors can help students quickly settle into their project work.
Social	4	A large amount of time in the beginning of the semester was spent on familiarizing ourselves using EDN as a tool. This potentially cut down on the time that could be spent on work reaching beyond semester goals. Chatbot can help future students achieve more in the limited amount of time they take Capstone. This can also streamline course by providing insight in what students use chatbot for, generating data that can be used to shuffle course material to fit each different project.
Environmental	2	Computation resources are limited, Virtual Machine that can serve the need of multiple sections are considerable beefier can what we can manage to acquire. Although this cost can be justified by the reduced load of EDN server which will see less students hunting around for information they need.
Economic	3	Automation is always more efficient when it comes repetitive tasks. Which saves instructors' time. Time that can be spent on answering more engineering related questions which in turn can help project progress faster. Counting utilities in economics terms, the chatbot helps instructors maximize the utility of their time.

## 15 Appendix E: Instructor Comments on Rule Generation Tool

This thread describes my experience in adding nodes (information) to the Chatbot. The JSON file is attached.

Reading the Instructions in [https://designlab.eng.rpi.edu/edn/projects/dl-chatbot-ta/wiki/Chatbot\\_Rule\\_Generator](https://designlab.eng.rpi.edu/edn/projects/dl-chatbot-ta/wiki/Chatbot_Rule_Generator) (5 min)

- The instructions were unclear.
- I expected that trying to a node clarifies the node creation process. However, it was not the case.
- I wanted to see an example.
- I wanted to see screenshots.

The first attempt to add node (30 min)

- It was unclear I had to download the JSON file from the wiki page.
- I recommend linking the JSON file for automated download. Syntax: [export:some/file](#)
- It was unclear how I could define a user's question.
- It was unclear if I had to update the parent node.
- Unique node IDs must be automatically generated.
- Linking a node to a parent node is difficult.

Find and organize information – 10 min to find the information in EDN. 20 min to find the additional information sources.

- How do I fabricate parts?
  - Your design must be reviewed and approved by your project engineer before fabrication. You must prepare reviewable designs, such as engineering drawings and circuit schematics. The guidelines are in [https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/Diagrams\\_and\\_CAD](https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/Diagrams_and_CAD)
  - You can ask the Design Lab Electromechanical Technician to fabricate parts using CNC machines. This information is not in EDN.
  - You can use the Doug Mercer '77 Laboratory, JEC 6033, for electronics. This information is not in the EDN.
  - You can fabricate parts by yourself in the Design Lab fabrication shop, JEC 2332. You must follow the safety guidelines <https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/Safety>. Regular shop hours are posted on the shop door. Typically, these are 8 AM - 4 PM, Monday through Friday. All other shop access requires. For more information, see [https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/After\\_Hours\\_Policy](https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/After_Hours_Policy)
  - You can also use the JEC Student Machine Shop (JEC 1010).
  - You can use RPI's fabrication services. [https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/RPI\\_Fabrication\\_Purchasing](https://designlab.eng.rpi.edu/edn/projects/capstone-support-dev/wiki/RPI_Fabrication_Purchasing)
- What equipment is available to use for my project?



- I can think of two types of equipment: Fabrication and Tests. How do I create nodes?
- The Design Lab Fabrication Shop has the following equipment: ??? We need to create a wiki page that describes the equipment available in JEC2332.
- The JEC Students Shop has the following equipment manual mills, manual lathes, laser cutter, CNC router, band saw, shear, and associated tooling.
- The equipment available in the Doug Mercer '77 Laboratory:  
<https://sites.ecse.rpi.edu//mercervlab/addingDatasheets.html>

### **My Observations**

- I did not find some necessary information on the EDN wiki pages. Therefore, adding new wiki pages and updating some wiki pages is essential.
- I had to look for appropriate Web pages in RPI. In other words, TA might not know enough to create nodes.