Drexel Chatbot Requirements Specification

Hoa Vu <htv27@drexel.edu>
Tom Amon <tpa27@drexel.edu>
Daniel Fitzick <dwf35@drexel.edu>
Aaron Campbell <ajc382@drexel.edu>
Nanxi Zhang <nz66@drexel.edu>
Shishir Kharel <sk3432@drexel.edu>

Version: 1.1

1. Revision History

Name(s)	Date	Comment(s)	Version
Daniel Fitzick	11/6/16	Preliminary Document Structure	0.1
Everyone	11/8/16	Filled in Function Requirements	0.2
Everyone	11/15/16	Filled in Remaining Sections	0.4
Everyone	11/19/16	Revisions of Introduction and Description	0.5
Everyone	11/20/16	Revisions of Functional Requirements	0.6
Everyone	11/21/16	Final Revisions	1.0
Everyone	12/4/2016	Feedback was reviewed and either incorporated or dismissed.	1.1

2. Introduction

2.1 Purpose of Document

This document will provide all of the requirements for the project Drexel Chatbot. It will serve as a reference for developers and customers during the development of the final version of the system.

2.2 Project Scope

Drexel Chatbot (Drexel natural language query service) is an AI chatbot that receives questions from users, tries to understand the question, and provides appropriate answers. It does this by converting an English sentence into a machine-friendly query, then going through relevant data to find the necessary information, and finally returning the answer in a natural language sentence. In other words, it answers your questions like a human does, instead of giving you the list of websites that may contain the answer. For example, when it receives the question "What time does the gym close today?", it will give a response “The gym closes at 10pm today.”

The main objective is creating a Web API, and sample web, mobile, and text messaging interfaces that demonstrate the use of the API.

The goal is to provide Drexel students and faculty a quick and easy way to have their questions answered, as well as to offer other developers the means to incorporate Drexel Chatbot into their projects.

2.3 Overview of Document

1.Revision History: Provide the date of, reason for, and people who were involved with the modification of this document.
2.Introduction: Provide an overview of the application, explain the objectives and goal of the project and describe the document structure.
3.Overall Description: Provide the specification of the system model, the classes model and the main constraints.
4.Functional Requirements: Provide the analysis of the requirements by feature.
5.Non functional requirements: Provide some other constraints that affect performance, safety and security.
6.Use Cases: Provide possible scenarios where the user interacts with the Web API and sample applications.
7.Glossary: Definitions of terms used.

3. Description

3.1 Product Perspective

Most of the search engines today, like Google, use a system (The Pagerank Algorithm) to rank different web pages. When a user enters a query, the query is interpreted as keywords and the system returns a list of highest ranked web pages which may have the answer to the query. Then the user must go through the list of webpages to find the answer they are looking for. Drexel Chatbot, however, will try to understand the query and provide a definitive answer.

There will be four main units to the system working together to understand the question and return an appropriate answer:

- Generic question construction - capable of taking a natural language question and making it more generic.
- Generic answer construction - capable of taking a generic question template and providing a generic answer template.
- Generic answer population - capable of taking a generic answer template and populating it with information from the database to form an answer.
- Information extraction - capable of finding information through structured or unstructured websites, and storing that information in a database.

3.2 Product Features

The major features for Drexel Chatbot will be the following:

- **Web API:** An API call will include a question in the form of a query string url parameter and the service will reply in JSON.
- **Natural Language Processing:** The system will take in questions written in standard English.
- **Natural Language Responses:** The answer to the question will be written in standard and understandable English.
- **Information Extraction:** There will be a database containing all the information needed, populated using information extraction techniques.

3.3 User Classes and Characteristics

The two classes of users for this system are described below:

- **API users**
  API users consist of application developers who want to incorporate Drexel Chatbot API into other software applications.

- **Mobile app/Web app/SMS users**
  These users consist of non-technical users who want to get answers for their questions. These users ask questions and get answers with mobile, web, or text messaging interfaces. This class of users include Drexel's current and prospective students, teaching faculty, and staff.

3.4 Constraints

3.4.1 Limited Question Scope

Creating a chatbot able to answer every single question about Drexel is not possible to implement with current technology and within the duration of the project, so the system will be able to answer questions about limited topics.

3.4.2 Language

The system will only support questions in standard English.

3.5 Assumptions and Dependencies

Keras is a library for creating and using neural networks. It should provide us with all the functionality we need, however if it is in someway deficient, then it will be replaced with a different library.

BeautifulSoup is a library for parsing HTML documents. It should be all we need to extract text from a webpage, but may be replaced if necessary.

We will develop the project using Python and MySQL database.

3.6 Requirements Apportioning

Each feature will be assigned an importance value. The project will be complete if all the features of Priority 1 and at least 50% of features of Priority 2 are implemented. No Priority 3 requirements are necessary.

Priority	Meaning
1	Essential, the project will not work without this feature. This feature will be implemented.
2	Important, the scope of the project will be significantly hindered without this feature. This feature will likely be implemented.
3	Desired, this feature complements the core functionality. This feature will be implemented, time allowing.

4. Functional Requirements

4.1 API Calls

4.1.1 Client Responsibilities

R4.1.1.1 The client will send a GET request to the Web API with the question as a URL parameter. **Priority 1**

R4.1.1.2 The client will specify the header Content-Type: application/json in their requests as convention. **Priority 1**

R4.1.1.3 A valid API query is a single URL parameter containing one sentence that is a question in standard English. **Priority 1**

R4.1.1.4 The server will reply with either data or an error, see R4.1.3.5. The client will be able to parse the JSON and determine if there was an error. **Priority 1**

4.1.2 Server Responsibilities

R4.1.2.1 The server will send all API data in JSON response documents with the header Content-Type: application/json. **Priority 1**

R4.1.2.2 The server will respond with a 200 OK status code if a request has the header Content-Type: application/json and is a valid API query. **Priority 1**

R4.1.2.3 The server will respond with a 400 Bad Request status code if a request does not specify the header Content-Type: application/json OR is a malformed API query. **Priority 1**

4.1.3 Response Document Structure

R4.1.3.1 API responses are defined in JSON, specified by [RFC 7159]. **Priority 1**

R4.1.3.2 A JSON object will be the root of every API response. **Priority 1**

The response document will contain at least one of the following top-level members:

R4.1.3.3 Data: the document's "primary data," in this case, the response to the client's query. **Priority 1**

R4.1.3.4 Errors: an array of error objects stating what went wrong with the client's request, should any issues arise. **Priority 1**

R4.1.3.5 The top-level members specified in R4.1.3.3 will not coexist in the same JSON document. If data is present, errors will be absent and vice versa. **Priority 1**

4.2 Generic Question Construction

4.2.1 Input & Output Format

R4.2.1.1 This unit will receive a text string from the URL parameter. **Priority 1**

R4.2.1.2 This unit will identify important words in the sentence and replace them with generic representations preceded by an escape character. (example: "PISB" becomes "$building") **Priority 1**

R4.2.1.3 This unit will output the sentence as a string, as described in R4.2.1.2. **Priority 1**

R4.2.1.4 This unit will output a map of generic representations to the words they replaced. **Priority 1**

4.2.2 List of Generic Representations

R4.2.2.1 This unit will have a list of generic words related specifically to potential queries (examples: building, professor). **Priority 1**

4.2.3 Error Handling

R4.2.3.1 An error during this process means there was a problem parsing the sentence and creating the generic question. In this case, return a message such as "Sorry, I didn't understand that." **Priority 1**

4.3 Generic Answer Construction

4.3.1 Input & Output Format

R4.3.1.1 This unit will receive the output sentence from the Generic Question Construction unit as input (R4.2.1.3). **Priority 1**

R4.3.1.2 This unit will generate a generic answer sentence using the input. (example: “What time does $building close?” becomes “$building closes at $closetime.”). **Priority 1**

R4.3.1.3 This unit will output the generic answer sentence described in R4.3.1.2. **Priority 1**

4.3.2 Error Handling

R4.3.2.1 If there was an error here, then the unit failed to create a generic answer given a generic sentence. In this case, simply fallback to the error handling described in R4.4.2.1. **Priority 1**

4.4 Generic Answer Population

4.4.1 Input & Output Format

R4.4.1.1 This unit will receive as input a mapping from the Generic Question Construction unit (R4.2.1.4). **Priority 1**

R4.4.1.2 This unit will receive as input a generic answer from the Generic Answer Construction unit (R4.3.1.3). **Priority 1**

R4.4.1.3 This unit will query the database for data about the elements in the mapping. **Priority 1**

R4.4.1.4 This unit will replace the representations in the generic answer with data. **Priority 1**

R4.4.1.5 This unit will output the answer to the original question, as described in R4.4.1.3. **Priority 1**

4.4.2 Error Handling

R4.4.2.1 If querying the database did not provide an answer, the system will say that it does not have an answer and provide appropriate website link where the user could find the answer. Example message, “I don’t know when Rush Building closes, but here is the website of the building. http://drexel.edu/cci/about/our-facilities/rush-building/” **Priority 1**

R4.4.2.2 If the system could not find appropriate website associated with the question, the system will return a generic error message such as “Sorry, I couldn’t find an answer to that.” **Priority 1**

4.5 Information Extraction & Database

4.5.1 Database

R4.5.1.1 A Stardog database will be used to store all information required to answer questions. **Priority 1**

R4.5.1.2 The database is populated by the information extraction unit before the rest of the system is available, so that all information is readily accessible. **Priority 1**

R4.5.1.3 The database will use the generic representations from R4.2.2.1 as table names or column headers for easier retrieval of data. **Priority 1**

R4.5.1.4 The database will be updated periodically and the API will be unavailable during the update. **Priority 1**

4.5.2 Structured Input

These are highly organized data sources, such that including the data into our database is simple. Examples: Databases

R4.5.2.1 Structural data sources will have their data stored in our database. **Priority 1**

4.5.3 Semi-structured & Unstructured Input

Semi-structured data sources are data sources with some organization, but the structure is not rigid enough to assure easy extraction of data. Example: table on a website.

Unstructured data is data with no organization, so extracting information is very hard to do programmatically. Example: A paragraph.

Extraction of information from semi-structured and unstructured data sources can be handled in 3 possible ways:

R4.5.3.1 For data with enough structuring, web scraping will be used to programmatically extract all the data needed and store it in our database. **Priority 1**

R4.5.3.2 For totally unstructured information, AI libraries will be used to programmatically extract and store any information possible. **Priority 2**

R4.5.3.3 Data that is especially hard to extract, for whatever reason, will be manually extracted and added to the database by a developer. **Priority 3**

4.6 Supported Question Topics

The Web API will only handle questions from following topics without unexpected error:
R4.6.1.1 Drexel facilities' locations **Priority 1**

R4.6.1.2 Drexel facilities' schedules **Priority 1**

R4.6.1.3 Drexel staff's office locations **Priority 1**

R4.6.1.4 Drexel staff's contact information **Priority 1**

R4.6.1.5 Drexel staff's positions **Priority 2**

R4.6.1.6 Drexel academics policies **Priority 2**

R4.6.1.7 Drexel admissions policies **Priority 2**

R4.6.1.8 Drexel information technology policies **Priority 2**

R4.6.1.9 On-campus dining locations **Priority 2**

R4.6.1.10 On-campus dining hours **Priority 2**

R4.6.1.11 On-campus dining food types **Priority 2**

R4.6.1.12 On-campus Food trucks' general locations **Priority 3**

R4.6.1.13 On-campus Food trucks' hours **Priority 3**

R4.6.1.14 On-campus Food trucks' food types **Priority 3**

R4.6.1.15 Location of official events listed on Drexel website **Priority 3**

R4.6.1.16 Schedule of official events listed on Drexel website **Priority 3**

4.7 User Interfaces

4.7.1 Website and mobile application GUI

R4.7.1.1 The GUI will have a textbox that will accept inputs from a keyboard. **Priority 1**

R4.7.1.2 Text box will originally contain a suggestive text question, to guide the user to the format of an appropriate question. **Priority 1**

R4.7.1.3 The GUI will have a "Send" button which sends text from the textbox to the API when clicked. **Priority 1**

R4.7.1.4 The GUI will have a chat window displaying questions sent to the system and responses from the API. **Priority 1**

R4.7.1.5 The chat window will contain all questions and answers from the current session, with a scroll bar if all messages can't fit on the screen. **Priority 1**

R4.7.1.6 If there is a network issue, the chat window will display an error message. **Priority 1**

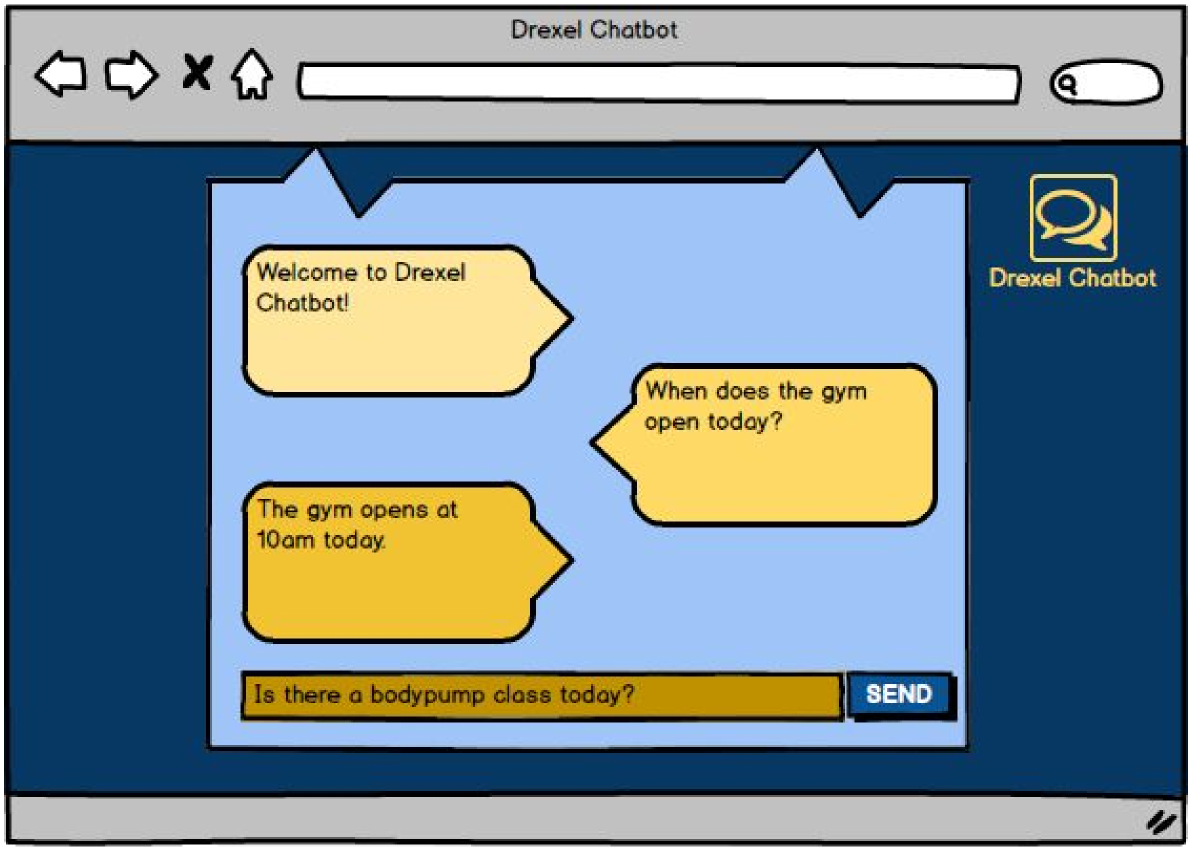

Drexel Chatbot
Drexel Chatbot
Welcome to Drexel Chatbot!
When does the gym open today?
The gym opens at 10am today.
Is there a bodypump class today?
SEND

Figure 1: Mockup of the Web user interface

4.7.2 SMS

R4.7.2.1 There will be a designated phone number that users can send texts to. **Priority 3**

R4.7.2.2 Texts sent to that number will be sent to the API, then the system will reply to the user with the answer as another text message. **Priority 3**

R4.7.2.3 If a question is not understood by our API, the system will send a text containing an example question after the text with the API response. **Priority 3**

5. Non-functional Requirements

5.1 API

5.1.1 Modularity

R5.1.1.1 The system will be designed in such a way that the algorithms for the four main units will be able to be easily swapped out. **Priority 1**

5.1.2 Accuracy

R5.1.2.1 The overall accuracy of the Web API's response will be measured using a developer-made testing set. **Priority 1**

R5.1.2.2 The overall accuracy is calculated by dividing total number of correct answers by the number of questions asked. **Priority 1**

R5.1.2.3 The accuracy of the Generic Question Construction part will be close to 80%. **Priority 2**

R5.1.2.4 The accuracy of the Generic Answer Construction unit will be close to 70%. **Priority 2**

R5.1.2.5 The accuracy of the Generic Answer Population unit will be close to 70%. **Priority 2**

The accuracy for each supported topic will be as follows:

R5.1.2.6 Drexel facilities' locations and schedules will have accuracy greater than 70% **Priority 2**

R5.1.2.7 Drexel staff's office locations, contact information, and positions will have accuracy greater than 70% **Priority 2**

R5.1.2.8 Drexel policies including academics, admissions, information technology etc. will have accuracy greater than 70%. **Priority 2**

R5.1.2.9 On-campus dining locations, hours, food types, etc. will have accuracy greater than 50%. **Priority 2**

R5.1.2.10 Food trucks' general locations, hours, and food types on Drexel Campus will have accuracy greater than 40%. **Priority 3**

R5.1.2.11 Official events listed on Drexel website, location and hours will have accuracy greater than 40%. **Priority 3**

5.1.3 Fast Response

R5.1.3.1 The average time for the server to respond, over the question testing set, will be less than or equal to 2 seconds. **Priority 2**

5.1.4 Security

R5.1.4.1 The connection between the Web API and the programs will use HTTPS, for security. **Priority 3**

5.2 Web interface/Mobile application

5.2.1 Ease of Use

R5.2.1.1 A new user will make less than 3 mistakes in 5 minutes after 5 minutes of use. **Priority 1**

6. Use Cases

6.1 Use Case Flow

6.1.1 API

Precondition: The server that is running the API is online.
Main Flow: The user sends their question as a URL parameter to our API's url.
Postcondition: The user receives the answer to their question in JSON.

6.1.2 Web/mobile application

6.1.2.2 Entering a question

Preconditions: The user starts the mobile application or web application.
Main Flow: The user enters the question in the text box
Postcondition: The text box will show the question entered.

6.1.2.2 Sending a question

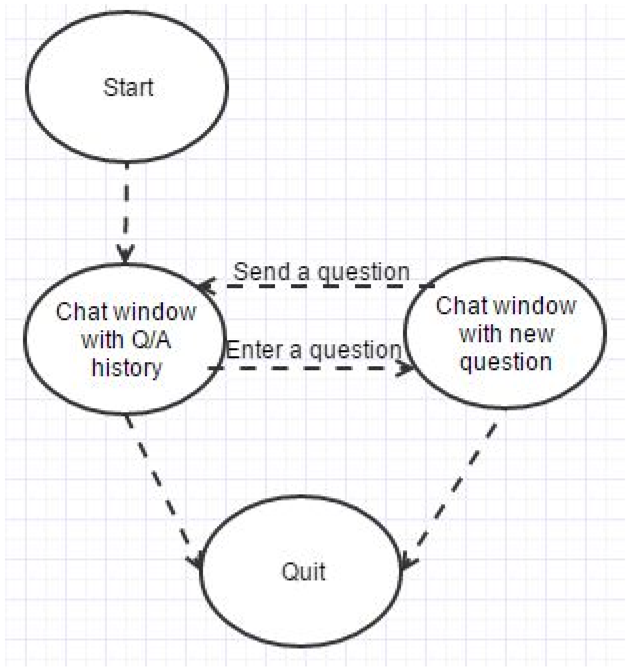
Precondition: Some texts exist in the Text Field box.
Main Flow: User clicks the send button
Postcondition: The texts in the Text Field box appear in the chat window, and the box is cleared out. After sometime, some texts generated by the software appear in the chat window.

6.1.3 SMS application

6.1.3.1 Sending a question

Precondition: User can send and receive SMS messages
Main Flow: Create and send a text message to the designated Drexel Chatbot number
Postcondition: The answer to the user's question is sent as a SMS to the number that sent the question.

6.2 Activity Diagram



7. Glossary

- Chatbot: An interface, usually text based, specializing in the mimicry of natural language conversation. AKA “artificial conversational entity.”
- GUI: Graphic User Interface, a type of user interface that allows users to interact with the software through graphical icons (e.g. buttons, etc.).
- HTML: Hypertext Markup Language, a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on webpages.
- JSON: JavaScript Object Notation, a data-interchange format that is commonly used in exchanging data over the Internet.
- Pagerank: PageRank is an algorithm used by Google to rank websites. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.
- SMS: Short Message Service, the text messaging protocol of cellular telephones.
- Standard English: the language that can be understood by English-speaking high school graduates.
- URL: Uniform Resource Locator, an address to a resource on the Internet.
- URL parameter: parameters whose values are set in a webpage’s URL.
- Web API: an application programming interface (API) for either a web server.
- Web scraping: web scraping is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database.