**Requirements Document**

**Anemone**

**REVISION HISTORY**

| Revision # | Author | Revision Date | Comments |
| --- | --- | --- | --- |
| 0.0 | Thomas Rowe, Wenkai Zhai, Faith Imadegbelo, Chenxuan Sun | January 22, 2022 | Project Design |
| 0.1 | Thomas Rowe, Wenkai Zhai, Faith Imadegbelo, Chenxuan Sun | January 25, 2022 | Requirement Documents done |
| 0.2 | Wenkai Zhai, ChunXuan Sun | TBD |  |
| 0.3 | Fay Johnston | January 30, 2022 | Expanded Use Cases, Revised abstract |
| 0.4 | Thomas Rowe | January 30, 2022 | Added System Block Diagrams, and another Use Case |
| 0.5 | Chenxuan Sun | January 31,2022 | Added System Overview |
| 0.6 | Fay Johnston | January 31, 2021 | Revised abstract, cd, and bkg |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[Document Overview 5](#_heading=h.gjdgxs)

[System Overview 6](#_heading=h.30j0zll)

[Abstract 6](#_heading=h.1fob9te)

[Conceptual Design 6](#_heading=h.3znysh7)

[Background 6](#_heading=h.2et92p0)

[System Block Diagrams 7](#_heading=h.tyjcwt)

[General Requirements 10](#_heading=h.1t3h5sf)

[Features and Requirements 10](#_heading=h.4d34og8)

[Functional Requirements 10](#_heading=h.2s8eyo1)

[Nonfunctional Requirements 10](#_heading=h.17dp8vu)

[Use Cases 11](#_heading=h.3rdcrjn)

[Resources 13](#_heading=h.26in1rg)

## Document Overview

The Requirements Specification document defines the features and requirements for Anemone.

* System Overview
* General Requirements
* Features and Requirements
* Use Cases

## System Overview

### Abstract

Anemone is an application that centralizes users' documentation queries. A user will be able to search for references without leaving their current application. For unknown queries, default results will direct the user to sites like Google or Stackoverflow. The system will automatically save all query results.

Our software is primarily designed to help people who are interested in programming. The utmost intention of the software is to make it easy for people to query the web without going down rabbit holes. Users can create personal accounts through which they can store information, and log in to the same account to display historically stored information if the device is changed.

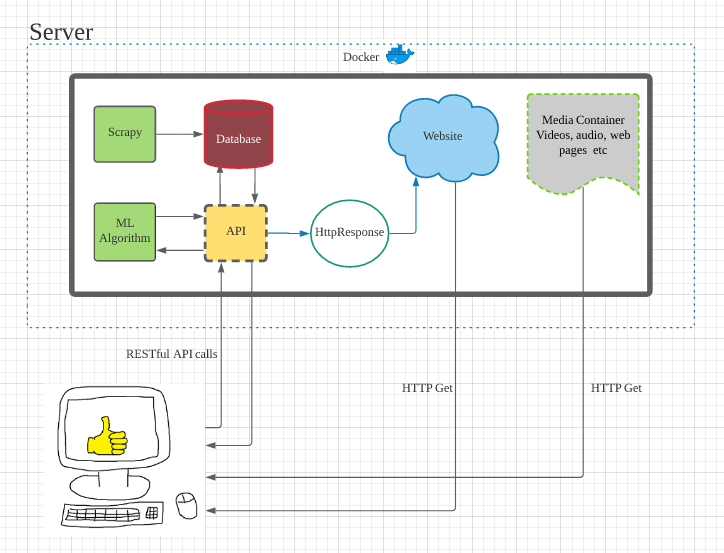
### Conceptual Design

Anemone will be designed as an iOS supported plugin. The front-end design serves to enrich the content, while the back-end will help to collect, organize, and parse the data.

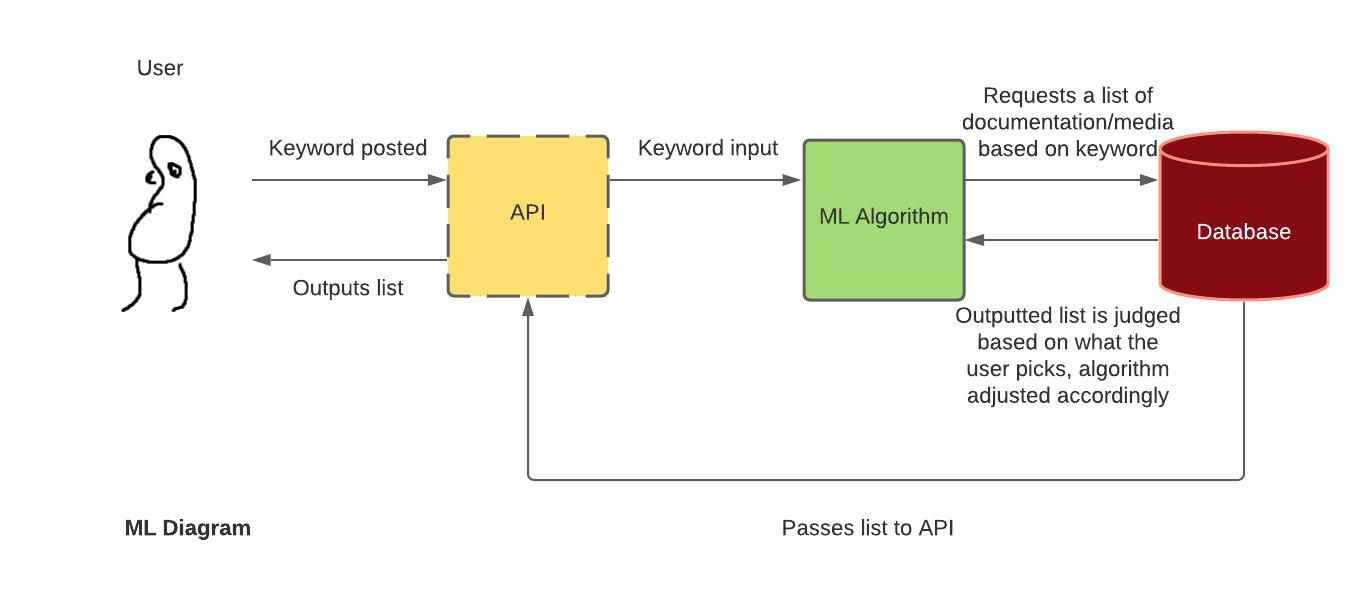
### Background

Anemone's main job is to help users shift left, reduce friction created when references are required, and increase productivity. Dash inspired the design for Anemone. The main difference between Anemone and Dash is that we distribute said documentation for free.

## System Block Diagrams

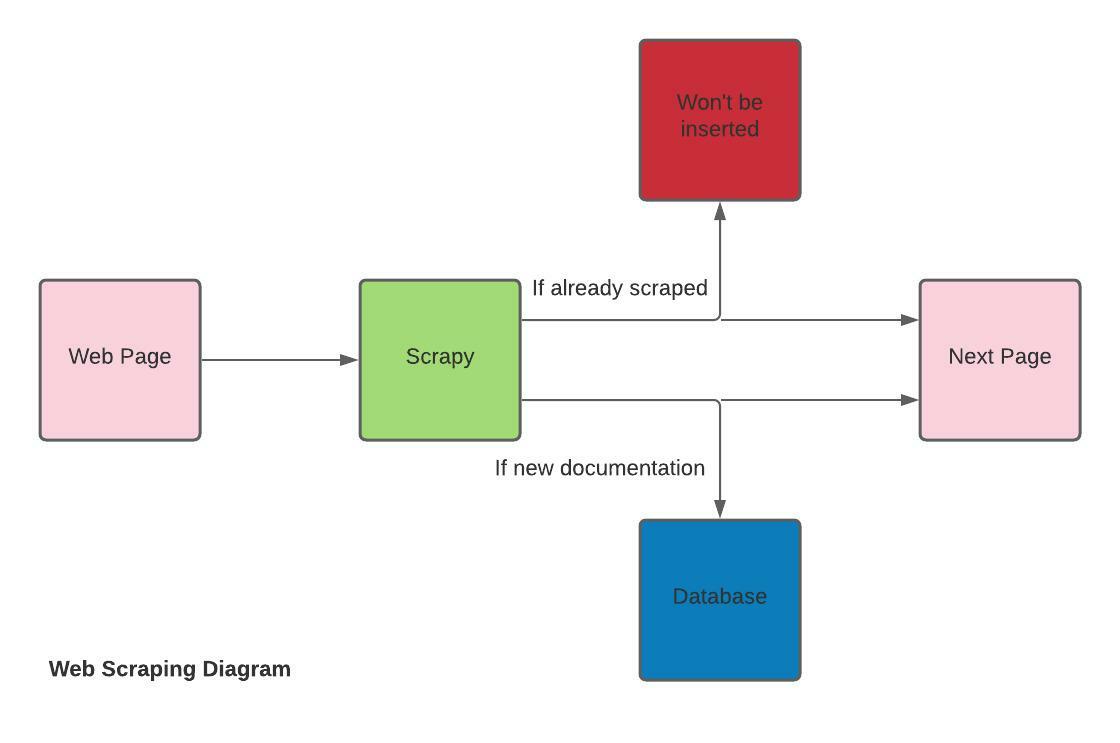
**Figure 1**. High-level design of the Anemone application.

In **Figure 1**, the diagram shows in a broad way how the user will be able to send requests and retrieve responses from the system. Essentially, the user will send API calls to the modular machine learning function, or if it’s not being used at the time directly to the database. Depending on where it’s sent, these calls will be able to retrieve a list of recommended articles/documentation in which the user can select from. Depending on the choice that the user makes, the machine learning algorithm will be updated accordingly, and the choice the user makes will load the documentation associated with the request.



**Figure 2.** Low level diagram of the machine learning function and its connections

The machine learning function’s connection with other functions is shown in more depth in **Figure 2**. Essentially, the user will type in a keyword in order to receive a response from the system. The way the system achieves this will depend on whether or not the machine learning function is currently being utilized. For instance, in the case that it is as described in **Figure 2**, the API will send that input directly into the machine learning model, in which it will decide what items specifically it should be looking for in the database, and once those items are found, the database will send a list to the user, and depending on which suggestion is picked, that will tweek the machine learning model accordingly.



**Figure 3.** Web scraping function

The web scraping function as depicted in **Figure 3**, will have a web scraper built in scrapy go from page to page looking for new documentation, video, and etc to scrape. In order to keep the system free from duplicates, everytime Scrapy scrapes something, we will need to compare the new page to what’s stored in the rest of our database. The simplest way to do this would probably be to store each piece of information with an associated keyword, whether that be the page’s header or the page’s URL. That way, the database will be kept relatively clean, and in the case of extracting a response from the database without the machine learning model, it would give us a way to search for data without one.

## General Requirements

* Google Chrome, Mozilla Firefox, or some other web Browser
* IDEs: Intellej, Pycharm, VScode etc.
* Server: Docker
* API: RESTful, React

## Features and Requirements

### Functional Requirements

* Anemone will offer users a tool to search for various tutorials.
  + The basic view page will show the default page and allow users to search for a specific field.
  + The side functionality is built for users to annotate and mark the importance of a page.
  + The Group tool shares the information within the group, and displays the comments from the group members.

### Nonfunctional Requirements

* Anemone will have a user-friendly interface on its webpage.
  + Anemone will not share any malicious links.
  + Display different error pages and redirect users to the original page.
  + Anemone will include the programming tutorials and academic resources for users to better understand a concept.
  + A list of previously viewed pages on the search bar as well as relevant pages to the current query.
  + To utilize Anemone better, go to the Anemone official website for more helpful references.
* Anemone will guarantee a high quality of security and privacy protection for users.
  + By creating private storage for every user, prevention of information leaking.
  + Anemone has algorithm checking, code certification, and mobile message confirmation to offer users a quick, safe, direct way of finding accounts back,
* Anemone will provide a great environment setting for users who need an excellent experience on searching.
  + Default Environment– Gather information from user’s input, searching asynchronously under the back-end
  + Quick Search Setting – Define users themselves first, allow us to deploy relevant contents first, then take the user’s searching.

## Use Cases

**Use Case 1**

A user wants to generate and document data collections

1. The user finds online resources he/she wants to save for future use and highlights it.
2. The user goes to Anemone web app on their personal computer and is greeted with a login screen.

*If the user needs to create a new account:*

1. The user clicks on the link to register and fills in the required information needed.

*If the user encounters an error entering in information (e.g., their password length is too short, email or username include invalid characters, etc.):*

i. The system returns an error message, prompting the user to change their credentials, going back to step 2a.

1. The system creates a new account with privileges tied to the user and the account.
2. The user is redirected to the homepage.
3. The user enters in their username and/or email, and password, and clicks login.

If the user enters incorrect login details:

1. The user is alerted that the information entered is incorrect and goes back to step 3.

4. The user is redirected to the homepage where they can find their highlighted text.

**Use Case 2**

A user wants to manage and search through their documentation pages.

1. The user navigates to their Anemone account and clicks on the search bar.
2. The user types in words related to the document to be search for.
3. The user clicks on the link provided by the search results.

*If the word/words is associated with documents in the users database:*

1. The links to data contained in the user's account will be displayed.
2. The user will click on a link and be redirected or choose to cancel the search.

*Else:*

1. A link to an external google search will be displayed.
2. The user will click on a link and be redirected or choose to cancel the search.

**Use Case 3**

A user wants to comment and annotate the use of documents found online

1. The user logs in to their profile and begins the process of saving the tutorial lines needed.
2. In the save window, the system gives the option to add comments to each highlighted text.

*If the user chooses to add a comment:*

1. The user types in a short description related to the document.
2. The user clicks on add comment.
3. The user clicks on save and their new document is added to the collection.

**Use Case 4**

A user wants to have automatic grouping of data

1. The user navigates to their data collection in their account.
2. The user wishes to search for multiple documents contained under a topic but does not want to individually search each document.
3. The user is provided with several tags which use machine learning to automatically group terms with similar functions.
4. The user selects the keyboard shortcut tag.
5. The user clicks the link and is provided with multiple saved documents pertaining to keyboard shortcuts.

**Use Case 5**

A student user using software needs to find tutorials. User wants documentation recommended as notification.

1. The student opens the app
2. Anemone is offered as an installable plugin
3. The plugin is activated (or deactivated)
4. The user resumes programming
5. With activated plugin, upon a hover event (on a resource), Anemone displays a mapped reference by tagging
6. If documentation is clicked, the reference is stored as a record for personalized history

**Use Case 6**

A user can search personalized documentation history

1. The student opens the app
2. Anemone is an installable plugin
3. The plugin is activated (or deactivated)
4. The user resumes programming
5. The user activates the search bar
6. Upon search bar event, personalized history store is retrieved
7. In a search bar, the user searches their history, by keyword

**Use Case 7**

A user wants to quickly and easily share large amounts of documentation with a team

1. The user logs into their Anemone account
2. The user either creates a group or joins a pre-existing group
3. When the user finds a new documentation page they wish to save and share with their group they
   1. Add in comments/annotations as to why the page is important and which parts of the page are important
   2. Save the page to their group where they and other group members can access it at a different time

## Resources