Development Plan

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 | Faith Imadegbelo | Jan 30, 2022 | Added use case 1-4, revised system overview |
| 0.4 | Fay Johnston | January 30, 2022 | Expanded Use Cases, Revised abstract |
| 0.5 | Thomas Rowe | January 30, 2022 | Added System Block Diagrams, and another Use Case |
| 0.6 | Chenxuan Sun | January 31,2022 | Added System Overview |
| 0.7 | Fay Johnston | January 31, 2021 | Revised abstract, cd, and bkg |
| 1.0 | Chenxuan Sun | Feb 1, 2022 | Created document layout |
| 1.1 | Fay Johnston | Feb 2, 2022 | Filled out every section and striked text that needs additional buy-in |
| 1.2 | Fay Johnston | Feb 3, 2022 | Add dates to Milestone Demos |
| 1.3 | Faith Imadegbelo | Feb 6, 2022 | Added burndown chart |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[Document Overview](#_gjdgxs) **4**

[System Overview](#_30j0zll) **5**

[Abstract](#_1fob9te) 5

[Conceptual Design](#_3znysh7) 5

[Background](#_2et92p0) 5

[System Block Diagrams](#_tyjcwt) **6**

[Activities](#_3dy6vkm) **9**

[Requirements Gathering](#_1t3h5sf) 9

[Functional Requirements](#_43siiab9c6ow) 9

[Nonfunctional Requirements](#_z6njg9iqrs98) 9

[Top-Level Design](#_2s8eyo1) 10

[Detailed Design](#_17dp8vu) 10

[Test](#_kr0auir1ke29) 11

[Tasks](#_26in1rg) **12**

[Schedule](#_35nkun2) **14**

[Development Environment](#_1ksv4uv) **15**

[Version Control](#_44sinio) **16**

[Feature List for Demos](#_2jxsxqh) **16**

## Document Overview

The Development Plan Document specifies the following:

* System Overview
* Activities
* Tasks
* Schedule
* Development Environment
* Version Control
* Feature List for Demos

## 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.

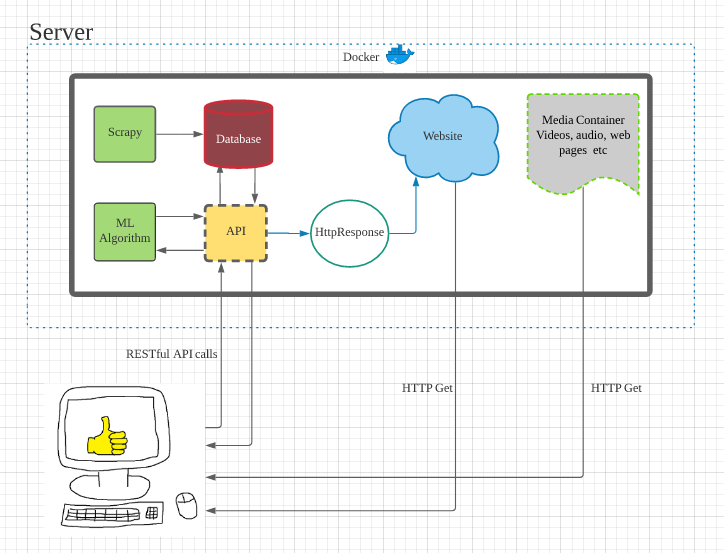
### Conceptual Design

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.

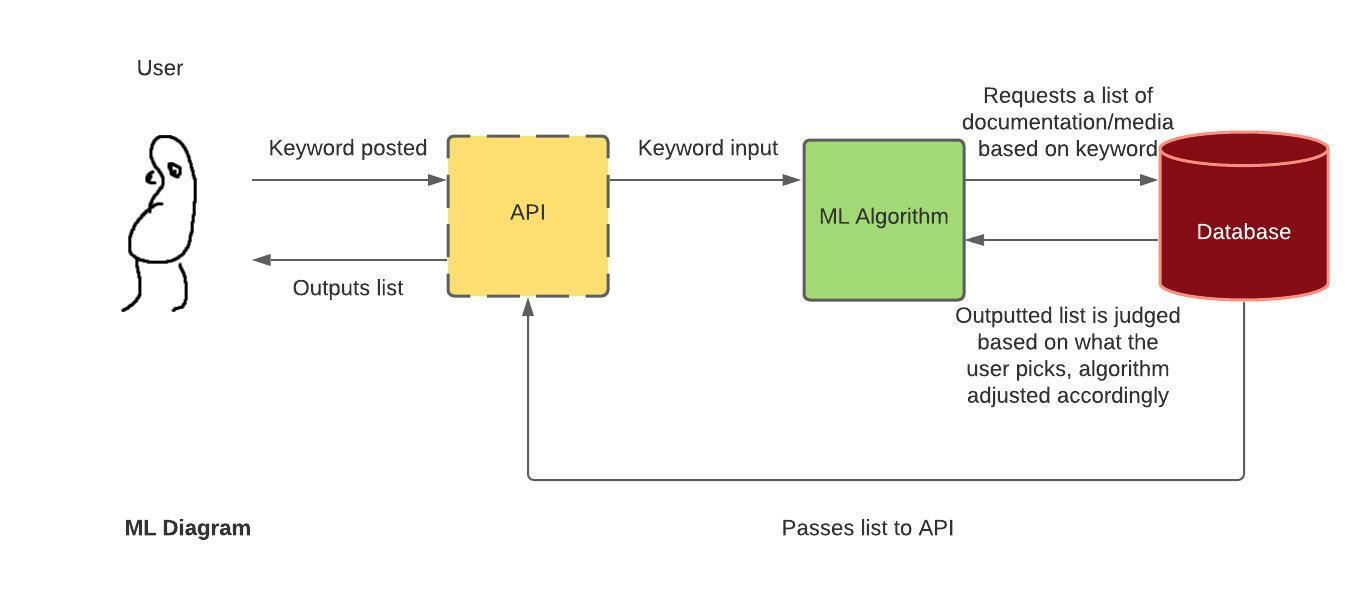
### Background

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

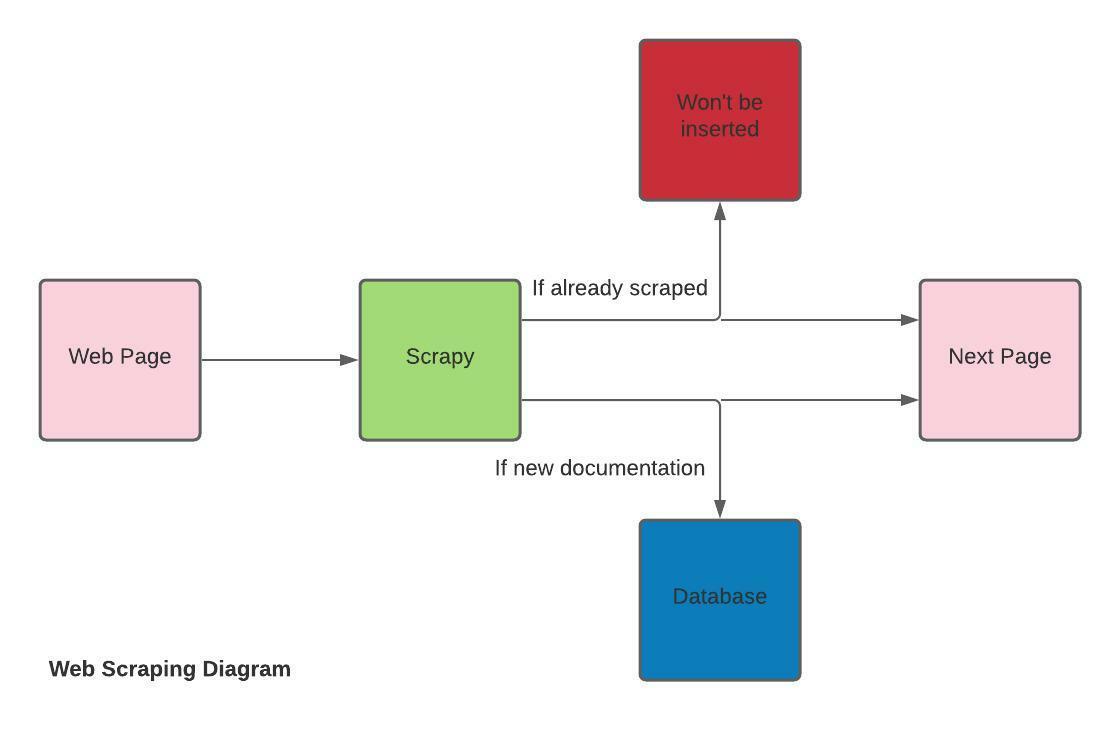
**Figure 1**. High-level design of the Anemone application.

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.

## Activities

### Requirements Gathering

### 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.

These application requirements were gathered based on several factors. Personal research and consultation with James Applebaum led to a base set of goals and functionalities. This included analyzing existing applications such as Dash and deciding how Anemone would be different. Further refinements were then set out by integration research. For example, when dealing with a user’s portfolio, there had to be a pre-defined authentication flow for user registration. This list was finalized after reviewing several open source software packages and various free technologies.

If time prevails, the User experience requirements gathering will be based on iterative prototypes that users can assess. This will be done by conducting User Acceptance Testing, asking the user to either execute certain functions, or follow an entire use case provided. Feedback and responses to the user’s tests would be recorded through an Excel sheet or a Google Form. Changes would be made as necessary to facilitate user flow in the application’s functions.

### Top-Level Design

1. Create a module to perform cluster analysis on a Search Behavior data set.
2. Create a Machine Learning Classifier Model to categorize queries into keyword categories.
3. Implement user account management for the web application to allow for user accessibility.

### Detailed Design

1. **Create a module to perform cluster analysis on a Search Behavior data set.**
   1. Design a questionnaire for investing behavior attributes.
   2. Create a data set using investing behavior and user portfolio attributes.
   3. Test different clustering algorithms.
   4. Create an interface for the clustering algorithm.
2. **Create a Machine Learning Classifier Model to categorize queries into keyword categories.**
   1. Research short text classification in similar areas like banking transactions.
   2. Create a suitable dataset for training the model.
   3. Train and Test different classifier models.
   4. Integrate the model with the website interface.
3. **Implement user account management for the web application to allow for user accessibility.**
   1. Design the user interface component.
   2. Allow users to create an account.
   3. Allow users to log into their account.
   4. Allow users to recover their account should they forget their account information.
   5. Implement and store the Advanced Setting in the user account database table.

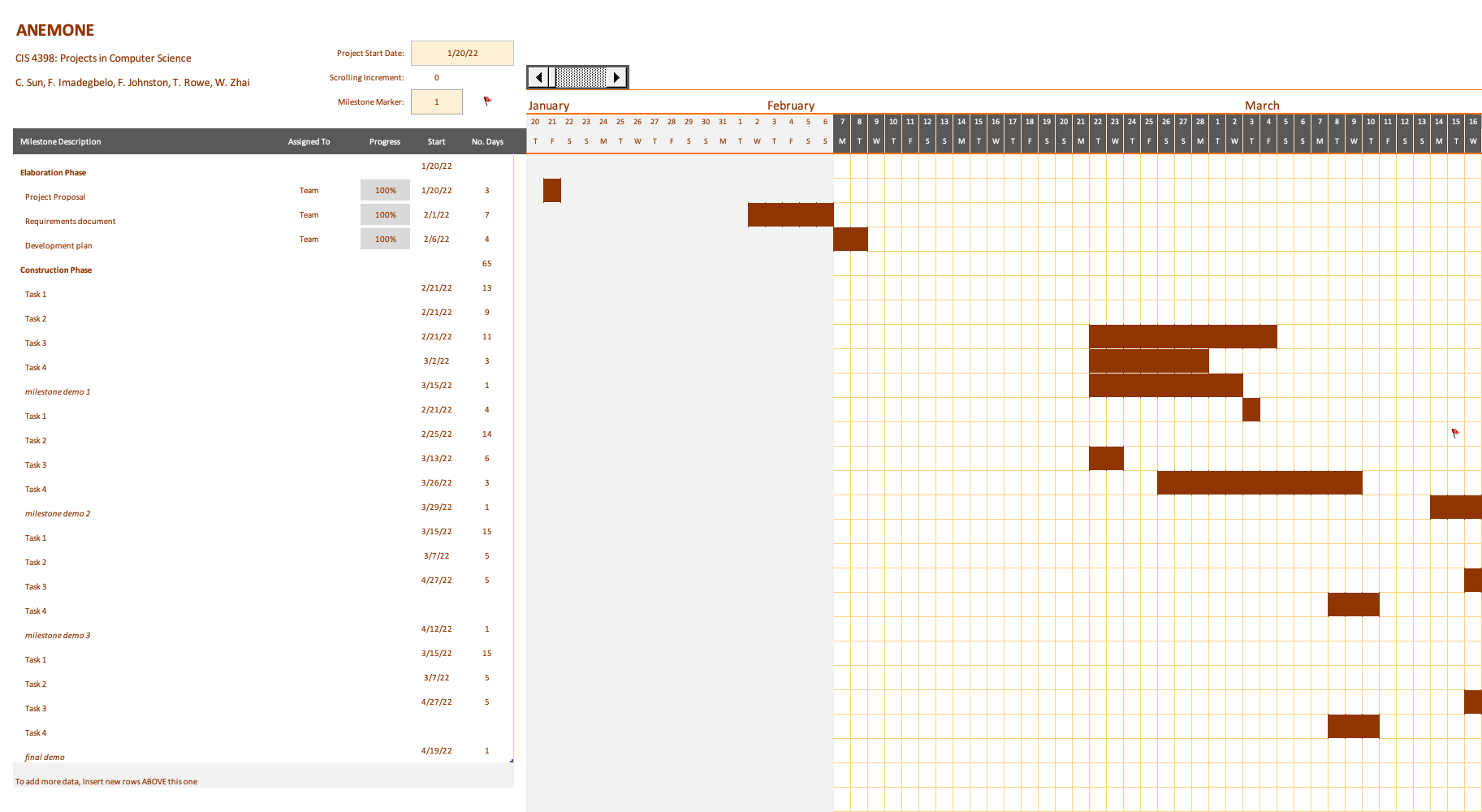
### Test

* UI Testing will be performed to ensure that the plugin’s features are functioning as expected. The user can input data pertaining to their personal finance, and the elements in the interface should not behave in unintended ways (i.e., processing urls when we should expect words).
* Unit Testing will be performed for the application to ensure that different logical units of code function as intended.
  + Unit testing for the Website will utilize
  + Server testing will be conducted via the Python unittest module.
* Integration Testing will be performed via making API Calls to the server’s endpoints, integrated with the plugin which makes the requests.
  + Automated API integration tests will be written using a web client to mock requests to the server.
  + Manual API integration tests will be conducted using Postman to issue HTTP requests to the server.
* Use Case Acceptance Testing will be performed to ensure that features that different users want are functional and match the users’ expectations.

## Tasks

1. **Create a module to perform cluster analysis on a Search Behavior data set.**
   1. Design a questionnaire for search behavior attributes.
      1. Define relevant data points.
   2. Create a data set using search behavior and user portfolio attributes. *(predecessor 1.2)*
      1. Define a model for search behavior data.
   3. Test different clustering algorithms.
      1. Import search behavior dataset to clustering environment.
      2. Run clustering algorithms with varying hyperparameters.
      3. Report results and select the best choice algorithm.
   4. Create an interface for the clustering results.
      1. Create clustering results API.
      2. Write unit tests to verify clustering results.
2. **Create a Machine Learning Classifier Model to categorize queries into keyword categories.**
   1. Research short text classification.
      1. Search public and academic sources for relevant data sets.
      2. Performance initial EDA of any data set of interest.
   2. Create or adapt a suitable dataset for training the model.
      1. Combine multiple datasets from 2.1 (if multiple sets are found).
      2. Clean the resulting data set and remove unnecessary attributes.
      3. Save the resulting data set for training and testing.
   3. Train and Test different classifier models.
      1. Split the data from 2.2 into training and testing sets.
      2. Train and test a variety of classifiers while saving the performance information associated with them.
      3. Identify the model that has the best balance of performance to resource usage.
      4. Save the model.
3. **Implement user account management for the web application to allow for user accessibility.**
   1. Design the user interface component.
      1. Plan out the general skeleton of the user interface.
      2. Implement the skeleton of the user interface with HTML.
      3. Utilize CSS to stylize the user interface.
   2. Allow users to create an account.
      1. Create a database table for managing account information.
      2. Develop a user registration form to handle account creation.
      3. Process user information through our API to save the credentials to a database.
   3. Allow users to log into their account.
      1. Create a form on the plugin that allows users to input their email and password.
      2. Obtain user account information related to the email being entered (done in the backend).
      3. Have a check that ensures an error message returns saying that an account associated with an entered email (one without an account) is invalid.
      4. If the account associated with the email is returned, compare the password entered with the password stored in the database (done in the backend), and authenticate the account if the passwords match.
      5. If the passwords do not match, return an error message stating invalid password.
   4. Allow users to recover their account should they forget their account information.
      1. Implement Email recovery utilizing a Python REST Auth module.
      2. Incorporate Email recovery into the website’s interface.
      3. Test out Email recovery feature.

## Schedule



## Development Environment

Editors and IDEs:

* VS Code
* PyCharm
* Anaconda
* IntelliJ

Languages, Frameworks, and Management Systems:

* Python
  + requests
  + Pip & Pip3 (Module Management System)
  + Data Science Modules
    - Pandas
    - BeautifulSoup
* JavaScript
  + NPM (Node Package Manager)
  + jQuery
* HTML
* CSS
  + Bootstrap (CSS Framework for Interface design)
* Database
  + TBD

Compilers and Interpreters:

* Python 3
* NPM

Documentation tools:

* Markdown

Testing Tools:

* Python Testing Framework (unittest module)
* Postman

Deployment Platform:

* TBD

## Version Control

* GitHub will be used as the version control system for Anemone.
* Branch protection will be utilized to prevent direct pushes to the main branch (at least one review required). This ensures that the main branch always remains a safe, reliable, and deployable codebase.
* Thorough code reviews will be done to verify any changes. Changes themselves should be well-described in commit messages.
* Each task/feature will have its own branch for the group to work on.
* Branches will be kept up-to-date and merged with the Main branch on a consistent basis, in order to minimize the occurrence of merge conflicts.

## Feature List for Demos

#### Milestone Demo 1 - Tuesday Mar 15, 2022

* **F1 User Account Management**
  + R1.1 User can create an account provided an email, a username, and a password.
  + R1.2 Users can access the Anemone plugin via entering in their credentials.
  + R1.3 Users can log out of the plugin upon accessing the pop-up.
* **F3 Interface Overview**
  + R3.1 Users can toggle showing or hiding the sidebar.
  + R3.2 Users can expand the different components of the sidebar to see an expanded menu of provided tools.
  + R3.3 Users attempting to access a tool in development are accompanied by a warning stating that the link is inaccessible rather than breaking the page.

#### Milestone Demo 2 - Tuesday Mar 29, 2022

* **F1 Plugin Overview**
  + R1.1 Users can search using the UI.
  + R1.2 Error messages will be displayed in a modal should you encounter any issues with your input data.
* **Teaser - Search Overview**
  + Showcase the Search Overview as a prototype component.

#### Milestone Demo 3 - Tuesday Apr 12, 2022

* **F2 Search Overview and Visualization**
  + R2.1 Interface demonstration (Searching Items).
  + R2.2 Visualization demonstration of the Search Categories.
* **Teaser - Machine Learning for the Keyword Classifier**
  + Demonstration of the progress of the Machine Learning Keyword Classifier, demonstrating success cases and failures.