SiteSearchCLUE   
Final Report

# 

Mumtahan Firdous, Trevor Gruber, Gavin Worley,  
 Juan Varela Ortiz, Nick Rios, Josh McGrew

# 

**Table of Contents**

[**Summary**](#_250h7pgmh0vc) **2**

[**Future Work**](#_15zwce1cm9rq) **2**

[**Appendices**](#_s3xdful50g5i) **3**

[Appendix 1](#_nj5wl3s1kp15) 3

[CBN](#_bjuldxl9rbut) 3

[Appendix 2](#_4djhyifl6vkw) 5

[SPMP - Software Project Management Plan](#_epfb8d3t4qcd) 5

[Feasibility Report](#_icp9xb2tle78) 13

[Appendix 3](#_u2snb2pztt7j) 15

[SRS - Software Requirement Specification](#_5y8tdg2iopb1) 15

[Appendix 4](#_xfhgux35kovo) 23

[SDD - Software Design Document](#_6kad69f4ktd5) 23

[Appendix 5](#_xcudharofxt9) 36

[STD (AT) - Software Testing Document (Application Testing)](#_injyx4riw839) 36

[Appendix 6](#_s1px19rbuzj) 45

[ATR - Accepting Test Report](#_9u3qw5o6nby2) 45

[Appendix 7](#_cc4ykryxtewh) 45

[User Manual](#_omn0driawzvl) 45

[Appendix 8](#_vp1kt1y88ah5) 50

[Expo Abstract](#_6q57h6ozomd9) 50

[Appendix 9](#_njsclztemgjd) 51

[Expo Poster](#_8f8pqd4ltw85) 51

[Appendix 10](#_n841qxmlgqd4) 52

[Demo Video](#_dsywbus480y9) 52

[Iteration Reports](#_x8mipmy53czp) **53**

[Iteration-1](#_tlfmgfc351b6) 53

[Iteration-2](#_f7elrhg3x3cw) 55

[Iteration-3](#_uli6ydgd7ma) 56

[Iteration-4](#_79mjgu3qd0gp) 59

[Iteration-5](#_4dfdn814w3j2) 61

[**Developer’s Profile**](#_1a3ctbwchoe9) **63**

# Summary

Over the course of the last 2 semesters, we developed a recommender system that recommends personalized search terms, based on the search histories of other, similar users. User similarity is calculated using the cosine similarity of users' past searches or collaborative filtering, this means that users are classified into neighborhoods according to the value returned from the calculation. The system uses this ranking data to recommend the search terms that similar users would “recommend” to the user. This system is demonstrated using a prototype website called Mini B College. This prototype was developed using Python (using the Flask, Pandas, and Surprise libraries), MySQL, Javascript (with JQuery), and HTML with CSS.

# Future Work

While the prototype does work, we would like to add more to it when we have time. One important mechanism would be a registration mechanism on the front end. As it is, while we do have code to register users on the back end, we don’t have a user-friendly front-end mechanism. We also plan to add a mechanism to recommend common keywords. We also are considering streamlining the UI, and plan to remove unnecessary tables to make the system more lightweight. Further, we plan to incorporate the software into real-world websites in various application domains and perform quantitative evaluation and we will investigate extensions such as hybridization approaches and diversified search term recommendations.

# 

# Appendices

## Appendix 1

### CBN

**Client:**

PSTogether (Personalized Software Together) - Prof. Dr. Young Park, Dept. of CS&IS, Bradley University (Would you join a start-up project together?)

**Product Description:**

Website search is important for a website’s success. Search can be characterized as either lookup (known-item) or exploratory (navigational) and precision-oriented or recall-oriented. Professional search such as legal search, patent trademark search, and evidence-based medicine search is recall-oriented.

Choosing the right search terms is crucial to a successful search, but is often difficult, especially for the exploratory search case.

Our goal is to guide visitors’ website searches, whether lookup or exploratory, by suggesting both non-personalized and personalized words that are potentially relevant to individual visitors and helping them select better search terms for their searches, whether precision-oriented or recall-oriented.

As an entrepreneurial start-up project at PST, we plan to develop (product idea validation, requirements, design, implement & test) a novel site search term recommender system called SiteSearchCLUE that provides both non-personalized search terms and personalized search terms within a website during the website search process.

Given a visitor, SiteSearchCLUE recommends both non-personalized and personalized clues for search terms as follows:

* Non-personalized search-term clue recommendation: (1) Search terms that visitors have entered most into the website search box & Keywords describing the web pages that visitors have visited most at a website and (2) All available keywords describing the web pages that are retrieved, not retrieved and both by a search engine at a given search step.
* Personalized search-term clue recommendation: Search terms that other visitors similar to the visitor have entered into the website search box & Keywords describing the web pages that other visitors similar to the visitor have visited using (1) the history of all visitors’ search terms and (2) the history of all visitors’ web pages via user-to-user collaborative filtering.

We will build a sample e-commerce/nonprofit website and develop a SiteSearchCLUE prototype for the website via RS tools such as Python, Surprise, Django, NLP keyword extraction tools such as TF-IDF, and SciKit-Learn and semantic textual similarity tools like Word2Vec and Doc2Vec. We will also explore experiments on effectiveness and usability.

SiteSearchCLUE helps website visitors browse and select better search terms by recommending non-personalized and personalized words as potential search terms.

We believe SiteSearchCLUE can be incorporated into any website and used to guide visitors to do an effective lookup and exploratory searches within a website!

**Technical Background:**

With the explosive growth of goods, services, and information available on the Web under the e-commerce umbrella, it has become increasingly difficult for consumers to find what they are searching for in a timely and efficient manner. Providing consumers with personalized recommendations helps them better find relevant and interesting products in the age of information overload.

Recommender Systems (RS) are software applications that provide users with personalized recommendations of potentially relevant and interesting goods, services, or information, and thus, help users navigate the information overload. The field of recommender systems is based on various technologies and is highly interdisciplinary. Though relatively new, recommender technologies have made significant progress in the last decade. A variety of recommender systems have been developed and successfully used in a variety of application areas, notably including e-commerce application domains such as Amazon, Netflix, Google, Facebook, and Youtube. One illustration is the famous Netflix competition, which offered a one-million-dollar prize in exchange for an algorithm to enhance the recommendation accuracy (i.e., movie rating prediction) of its recommender systems. Today, recommender systems have become a vital part of e-commerce websites.

Collaborative Filtering (CF), which is based on user collaborations, is the most widely used and proven method of providing personalized recommendations. A traditional recommender system – called a single-criterion rating recommender system – is based on user feedback (ratings) on items in a single criterion. For example, a traditional hotel recommender system uses the overall ratings of hotels given by users. A multi-criteria rating recommender system extends the traditional model by using user feedback (ratings) on items in multiple criteria instead of a single overall criterion and tends to show better recommendation accuracy. For example, a hotel recommender system based on multi-criteria ratings was proposed and evaluated.

## Appendix 2

### SPMP - Software Project Management Plan

**1.0- Introduction**

The end goal of this project is to create an internet application that searches websites and logs user searches to provide both personalized and non-personalized user search recommendations. Personalized user searches will help users find what they are looking for, and non-personalized search recommendations will also be given.

*1.1- Project Scope*

This project has to be usable with most websites and be able to recommend personalized and non-personalized searches. Non-personalized search recommendations are simply popular searches that are recommended to everyone, while personalized search recommendations are based on prior history compared with the histories of similar users.

*1.2- Major Software Functionality*

The software will be able to

* Allow users to register and log-in
* Search websites of different varieties with minimal setup
* Store searches of each user
* Find users who search for similar things
* Calculate popular searches, and recommend them to users as non-personalized searches
* Calculate what users search similar terms, and recommend personalized searches based on what similar people commonly search

*1.3- Performance/Behavior Issues*

The software will be user-friendly. It should be able to find search results that are relevant to what the user is searching for. It should be able to be used on most modern browsers.

*1.4- Management and Technical Constraints*

We have a fixed deadline of April 29, 2022. This is a drop-dead deadline. We retain the rights to the source code, so the deliverables will not include it.

*1.5- Feasibility*

The client wants to develop a search engine that provides both personalized and non-personalized search options for websites. From the feasibility report, we can see that creating this program is feasible as (1) it is possible to collect data from users/visitors and store them for further filtration, and (2) there have been recommender systems like this for other search jobs, such as Netflix for personalizing movie search and suggestion, (3) as this project will help user to search website in a timely and efficient manner using personalized recommendations, this project is feasible.

\*\*For additional information, see the [Feasibility Report](#_icp9xb2tle78) document.

**2.0- Project Estimates**

*2.1- Historical Data*

The SiteSearchCLUE team has collectively performed one similar project of this scale in the past. Utilizing the estimation data from this previous project, alongside the expeditious development timeline requested by the client, it is estimated that each software engineer within the SiteSearchCLUE team, will be dedicated part-time to this project. With each member of the SiteSearchCLUE team still being an undergraduate student, there are academic obligations that can hinder the ability of this team to be devoted full-time to the project. While many of the specific details of this previous project must be kept confidential, the average time required for the design, coding, testing, and release of this project was 39 days. The average projected price for the completion of this project was $7,040.

*2.2.1- Estimate Technique: Analogy*

Based on the project scope and client requirements, in comparison with the estimation data from the previous project mentioned in Section 2.1 (and taking into consideration both the experience and skillsets of the undergraduate software engineers and programmers at SiteSearchCLUE), it is estimated that this project should require seven man-months of labor. To satisfy the deadlines of the client, all seven members of our team will be collaborating on the project, from start to finish. This estimate of the amount of effort necessary to achieve the objective is based on the SiteSearchCLUE team's prior experience with this past project, which is analogous to the current, proposed project. SiteSearchCLUE will incur costs of $7,000 per man-month. These costs are broken down as follows:

*Hardware and Software Costs:* $90

Includes the expense of development tools (Integrated Development Environments, desktop computers, removable media, etc.).

*Travel and Training Costs:* $0

SiteSearchCLUE does not expect any travel or training associated with the completion of the project.

*Effort Costs:* $6,910

Includes employees' salaries, insurance, and facility overhead.

As a result, using analogies to prior projects, we estimate that the project should require 30 days, for $7,000. This result is reached as follows:

7 man-months / 7 people = 1 man-month (30 days)

1 man-month \* $7,000 per man-month = $7,000 project cost

*2.2.2- Estimate Technique: COCOMO II*

Per the COCOMO II model, the following weights will be used in estimating the proper number of object points for this project:

| Complexity Weight | | | |
| --- | --- | --- | --- |
| Object Type | Simple | Medium | Difficult |
| Internal Logical Files | 7 | 10 | 15 |
| External Interfaces Files | 5 | 7 | 10 |
| External Inputs | 3 | 4 | 6 |
| External Outputs | 4 | 5 | 7 |
| External Inquiries | 3 | 4 | 6 |

The progression of this project will be based upon component-based development, to ensure the efficiency and accuracy of the overall software. Therefore, SiteSearchCLUE estimates the number of internal logical files, external interface files, inputs, outputs, inquiries, and the corresponding number of object points, as follows:

Medium Internal Logical Files: 6

Simple External Interface Files: 4

Medium External Inputs: 8

Difficult External Outputs: 9

Difficult External Inquiries: 5

Number of Object Points (NOP) = (6 \* 10) + (4 \* 5) + (8 \* 4) + (9 \* 7) +

(5 \* 6) = 205

The developers at SiteSearchCLUE are equally experienced and merit novice ratings in both the experience/capability and environment maturity/capability categories, resulting in a "PROD" value of 29. The estimated effort is therefore calculated as follows:

Estimated Effort = NOP / PROD = 205 / 29 ≅ 7.07 man-months

This results in the following estimate for project cost:

7.07 man-months / 7 people = 1.01 man-months (31 days)

1.01 man-months \* $7,000 per man-month = $7,069 project cost

*2.2.3- Software Size Estimate (Lines of Code)*

Based on the prior experience of the developers at SiteSearchCLUE, it is estimated that this project will require approximately 900 lines of code.

# 

# 

**3.0- Risk Management**

| **Risk** | **Probability** | **Potential Impact** | **Mitigation, Monitoring, and  Management plan** |
| --- | --- | --- | --- |
| Insufficient resources | Low | Marginal | Sufficient resources should be deployed so the software can run smoothly |
| Database issues | Medium | Critical | The database must be designed correctly and relationships thoroughly audited |
| Change in project requirements | Low | Marginal | We will make adjustments if necessary |
| Incorrect keyword suggested | Medium | Critical | Thorough testing must be done before software deployment |
| User error | High | Critical | Software must be tested thoroughly and made to be user-friendly |
| Missed deadlines | Low | Critical | Proper project journaling, pacing, and division of work should prevent missed deadlines |

**4.0- Project Schedule**

*4.1- Process Model, Framework Activities, and Task Set*

The process model that best fits this project is the **incremental** model. During the project timeframe, increments will be developed and improved upon as needed.

*4.2- Framework Activities*

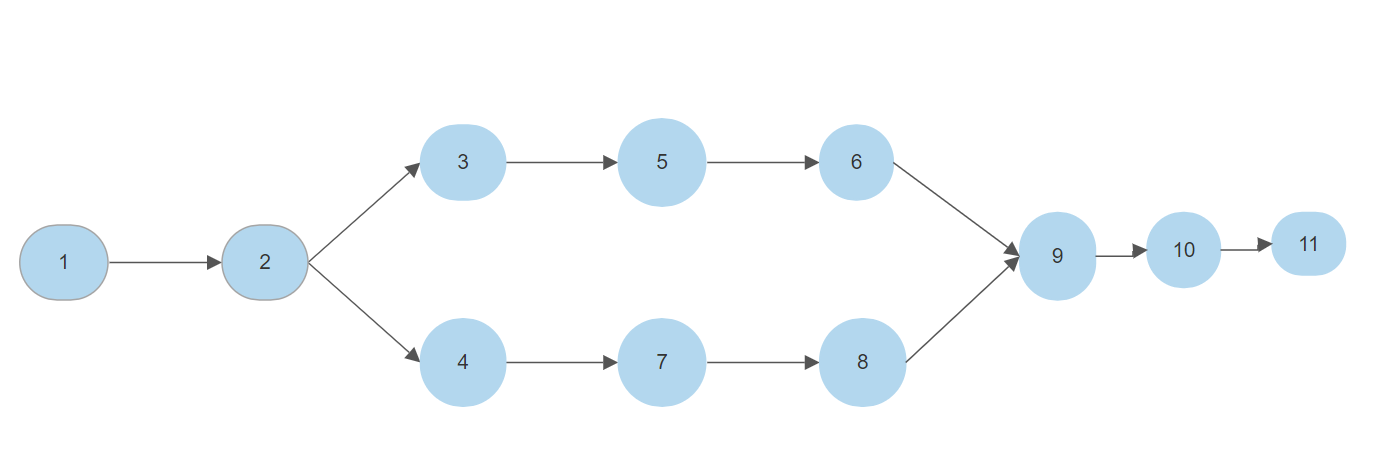
The framework activities appropriate for this project are:

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

*4.3- Task Set*

1. Project Scoping and Feasibility
   1. Milestone: Complete Software Project Management Plan
   2. Deliverable: Software Project Management Plan
2. Software Requirements
   1. Milestone: Complete Software Requirements Specification
   2. Deliverable: Software Requirements Specification
3. Front-End Software Design and Modeling
   1. Milestone: Complete front-end basic GUI model
   2. Deliverable: Software Design Document
4. Back-End Software Design and Modeling
   1. Milestone: Complete back-end model
   2. Deliverable: Software Design Document
5. User Interface Development
   1. Milestone: Complete UI coding
   2. Deliverable: Working UI
6. User Interface Testing
   1. Milestone: Complete UI testing
   2. Deliverable: Front-end testing results
7. Back-End Coding Functionality
   1. Milestone: Complete back-end coding
   2. Deliverable: Functioning back-end code
8. Back-End Testing
   1. Milestone: Complete back-end testing
   2. Deliverable: Back-end testing results
9. Front-End and Back-End Code Combination
   1. Milestone: Complete combination
   2. Deliverable: Functional UI and back-end code that works together to perform desired site searches.
10. Combined Code Testing
    1. Milestone: Complete combined testing
    2. Deliverable: Software Testing Document
11. Project Deployment
    1. Milestone: Release of a software application
    2. Deliverable: Software Implementation and Acceptance Test

*4.4- Task Network*

The network for the tasks are outlined below. Each number corresponds to an item in the task set in Paragraph 4.3.

*4.5- Timeline*

The tasks for this project are intended to be completed under the following timeline. The timeline is provided in Gantt chart form:

# 

# 

# 

**5.0- Staff Organization**

*5.1- Model*

Our team will operate using a Democratic Decentralized (DD) model. The “democratic” portion is appropriate because this structure allows the team to vote for a selected team leader. The communication is horizontal, so all team members will be communicating with each other. The “decentralized” portion is appropriate because we will be conversing with each member while also having a dedicated team leader.

*5.2- Roles and Responsibilities*

The primary roles and responsibilities for this project are as follows:

| **Roles/Member** | **Responsibilities** |
| --- | --- |
| Project Manager | Mumtahan Firdous |
| Quality Manager, Assurance/Testers | Trevor Gruber, Gavin Worley |
| Software Architects | Nick Stuart (graduated) |
| UX and UI Designers | Mumtahan Firdous |
| Front-end Programmers | Nicolas Rios, Josh McGrew, Mumtahan Firdous |
| Back-end Programmers | Gavin Worley, Juan Varela, Trevor Gruber |
| Database Admins | Trevor Gruber |

# 

**6.0- Tracking and Control Mechanisms**

*6.1- Software Quality Assurance*

1. SQA Plan: For quality assurance, the team will meet and review the software weekly. At this time, the team will discuss the technical reviews and talk about the steps that need to be taken to ensure the project meets its requirements and stays within the scope.
2. Technical Reviews: The team will review each integration of the software. The review will consist of changes made to the software which will reflect changes to the project plan. After reviewing the software, the team will review the project schedule, cost, and plan and update them as necessary.
3. Testing: The team will test the software’s code to find and remove issues. The main goal of testing is to ensure that the software meets its requirements, by eliminating the problematic code in the software.
4. Analysis: If after testing the code it is revealed that there are bugs, the team can analyze the code. The team will work together to discover where they are introduced, and possible methods to remove the errors. This will lead to a better understanding of the code and allow the team to critique the code to standards.
5. Audits: After testing the software, the team will gather the information and fill out a changes form that declares exactly what was changed and what it affected. This will allow the team to keep track of the progress of the software.

## 

*6.2- Change Control*

The primary focal points for change management and control are as follows:

1. **Identify Change:** The team will identify the changes that are made. These changes will be written down in a form that the team will fill out based on the way the change affected the software.
2. **Version Control:** The team will save versions of the software for purposes of reviewing, and tracking the progress, and will allow us to revert to an older version if something goes wrong during implementation.
3. **Report Changes:** The team will be reporting the changes made in a form that will describe what was changed and how it affected the program. This will allow the team to keep track of how the program changes. This will also establish traceability in our product, which will allow the team to know why every change was made.

## 

### Feasibility Report

**Client’s Need:**

* Characterized Search:
  + Lookup
    - known-items
  + Exploratory
    - Navigational
  + Precision-Oriented
  + Recall-Oriented
    - legal search
    - patent trademark search
    - evidence-based medicine search
* Clue Recommendation
  + Non-personalized
    - Search terms that visitors have entered most into the website search box & Keywords describing the web pages that visitors have visited most at a website and
    - All available keywords describing the web pages that are retrieved, not retrieved, and both by a search engine at a given search step.
  + Personalized
    - The history of all visitors’ search-terms
    - The history of all visitors’ web pages via user-to-user collaborating filters

**Technical feasibility:**

This search term recommender system is a search engine that collects data from visitors and visitors with similar interests and personalizes the search pattern so that it is easier for consumers to search for their needs promptly. It is possible to collect data from the users and store them according to their preferences and past web visits and when a similar word or category is typed, the user will be provided with filters and recommendations. One similar example can be Netflix, as they provide personalized and non-personalized movie options. So developing a project like this is possible to do over a year with reliable resources, examples, and management. So this SiteSearchCLUE project is technically feasible.

**Marketing feasibility:**

Other search engines also collect data and show suggestions while searching but for website search, there still haven’t been any popular filtering options that provide clues for the desired websites or sort in categories. So this seems to be a fairly new project on the market, which can be highly demanding because of the modern recommender system technology such as filtering options. So this project has a good chance of marketing feasibility.

**Financial feasibility:**

As this is a start-up project, and we students are working on it, there is currently no financial cost for this project. But if it is taken to the real world, this project might have not taken a whole year and would have cost between $50,000 to $250,000. Which is an average cost to build software within a year. So, this project is financially feasible.

**Social feasibility:**

This project is going to help website visitors browse and select better search terms by recommending non-personalized and personalized search terms. Other recommender systems have done similar things, and have not had many serious ethical debates, thus the project can be assumed to be socially feasible.

## Appendix 3

### SRS - Software Requirement Specification

**1.0 Stakeholders Identification**

Within the context of the CS490 course, SiteSearchCLUE is acting as the developers and Dr. Park as the client for SiteSearchCLUE. Beyond class scope, however, others with a stake in the project can be the business owners and website administrators that intend to implement the search capabilities of SiteSearchCLUE.

**2.0 Functional Requirements**

The following is our list of requirements for the SiteSearchCLUE software:

**FR-01**:  
The system shall allow the user to register an account for the service. This account will include a username and password, and other details are done via web form and can be accessed or changed later.

**FR-02**:  
The system shall allow the user to log into the system via the use of a username and corresponding password previously registered as an account.

**FR-03:**  
The system shall allow the user to enter strings of characters as searches to websites, and the system will search the website using the string.

**FR-04:**  
The system shall store the user-submitted searches for later use to compare with other users’ searches. It shall find similar searches, and be able to find which groups of users are most similar.

**FR-05:**  
The system shall use similar users’ searches to recommend personalized searches that the user is more likely to be interested in.

**FR-06**:  
The system shall ask/allow users to save their searches before logout.

**FR-07:**The system shall log out users without crashing or other behaviors indicative of a failure state.

**FR-08:**  
The system shall run without terminating itself or freezing unexpectedly.

**3.0 Non-Functional Requirements**

The following list contains the Non-Functional Requirements for the SiteSearchCLUE software. Each Non-Functional Requirement will be abbreviated "NFR" with the Non-Functional Requirements’ identifier at the end of the abbreviation:

**NFR-01:**

Performance - the average time to generate recommended search terms following user search input (a string of characters), shall have a baseline of 0.7 seconds, with a target time of 0.5 seconds. However, this time shall not exceed more than 1 second. These boundaries signify the overall performance rate of the main functionality of the SiteSearchCLUE software.

**NFR-02:**Availability - the system shall be available for use by users. The system shall be available to users for at least 167 hours per week, allowing for occasional unexpected outages.

**NFR-03:**Relevance - the search functionality shall return websites that are relevant to what the user searched. The system shall return results that users will consider relevant to what they searched for 75% of the time.

**NFR-04:**Usability - the system shall be easy to use. The simple search interface will allow for a minimal learning curve for users to achieve the desired results from the system. The expected performance rates of the system enable users to retrieve recommended search values efficiently. Familiar search icons and placement of the search field on the website give users the ability to return to the system interface with ease. Users may encounter errors in their recommended search values if the information in their user profiles does not reflect their interests or is not up to date. Users shall also be able to sign up for the service very easily. It shall be easy enough to sign up that within 3 minutes the average user will be able to sign up. This shall apply regardless of the browser the user is using.

**NFR-05:**Security - the system shall handle any exceptions and user input in the search field. The main vulnerability of this system is centered around the security of the database that will store and manage the user data, utilized for personalized search values. To mitigate the possibility of data breaches or losses, the system shall implement the Advanced Encryption Standard (AES) to encrypt the data on a column-level within the database. This symmetric algorithm will utilize a 192-bit block cipher encryption, which grants adequate security without hindering the rates for access time of the database information. Security of the software shall be measured by weekly testing of the encryption and privacy-protection techniques, by running simulated potential attacks on the system.

**4.0 Other Constraints**

The following is a list of other constraints that will be present with the SiteSearchCLUE system.

**OC-01:**

The system will be limited by the performance of any databases in use and their corresponding servers, including the maximum number of users on the server and query/search times.

**OC-02:**

The system will be limited by the software on which it is created and runs. This includes programming languages, design software, web browsers, and so on.

**OC-03:**

The perceived performance of the system may be limited depending on the website on which it is used. For example, a website may have a limited number of searchable items, and recommendations could be repetitive. Also, the website may not have what the user searches for and therefore the system may not make relevant recommendations.

**5.0 Use-Case Model**

**5.1 Actors**

* Users
  + A user is defined as a person who uses SiteSearchCLUE to get recommendations on searches. They shall be able to create an account, log in and out of their account, search the website, and receive recommendations. Upon viewing recommendations, they shall be able to determine if the recommended search is what they are looking for. The system shall have access to a list of their recent searches that will allow the system to recommend searches of similar users.
* Administrators
  + The administrators are the managers of both the recommender system and the search database. The administrator shall be able to log in and out of the system to manage it and shall be able to access the database at any time. They shall be able to add to the database and edit the information if it is needed.

**5.2 Use-Cases**

**UC-01:** User registering a new account in the system.

**UC-02:** User logging into the system

**UC-03:** User logging out of the system

**UC-04:** User searching the website using string

**UC-05:** User accessing a history of searches

**UC-06:** User saving searches

**UC-07:** User editing account details

**5.3 Use-Case Diagram**

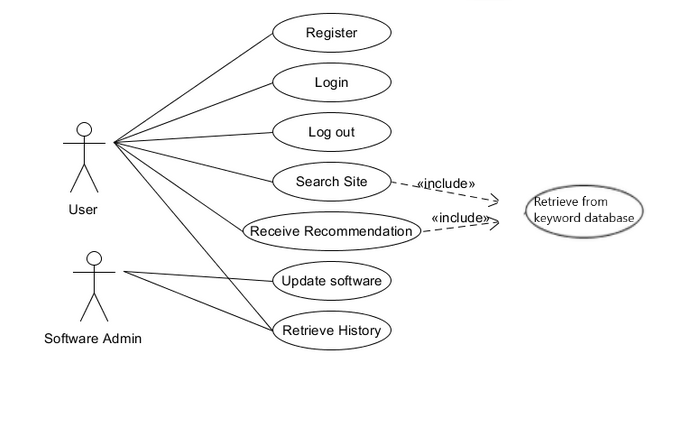


Image 01: Use-Case Diagram

**5.4 Use-Case Descriptions**

**UC-01**: User registering a new account in the system.

The user creates a new account with the service. He or she will visit the website and (as a new user) navigate to the account creation portal. Then, after submitting credentials, the user will receive a confirmation email to note creation was successful. The system will create a new account on the server, and it will be tagged with the proper role (administrator or general user).

1. **User action:** The user inputs a username, password, and email address to register with the service.
2. **System action:** The system verifies no existing accounts have the same username. If the new account passes the check, the system will store the username, password (after salting and hashing), and email address on the server. Sends a confirmation email to the user to confirm registration was successful.

**UC-02**: User logging into the system

General user or administrator logs into the service. The service will validate the user’s input to make sure it’s a valid account and correct password, then check for the account type. The system will display options specific to the account type, and allow the user to move through as he or she wishes.

1. **User action:** The user inputs a username or email address and password to log in with the service. The user will be greeted with options to manually create a database entry, fetch his or her history, update their profile, and general account information. Administrators will see tools to perform administrative tasks.
2. **System action:** Validates user’s input to attempt a login and displays applicable functionalities. Adds successful login to the login log.

**UC-03**: User logging out of the system

Logs out of service, disconnecting software from the current website. The user will use a standard logout button.

1. **User action:** Logs out.
2. **System action:** Close connection and end the user session.

**UC-04**: User search

The user performs a search query on his or her website of choice. The service shall read what the user writes to suggest keywords that may apply to the search. As the user types, the service will read the input and attempt to match it to existing database entries, then suggest words in real-time.

1. **User action:** The user begins a search
2. **System action:** The system will read the input in real-time, contact the keyword database, and suggest keywords that may be applicable based on the user’s past searches. Searches will be stored in the database and any accepted keywords shall be noted for future suggestions.

**UC-05**: Provide the user with a history of searches

Users can request the history of their searches, associated keywords, and other information stored in the database inside their profile. After logging in, the main menu will contain an option to request data. The user can select this option and a PDF file containing all the information requested will be sent to his or her email.

1. **User action:** Request data associated with his or her account.
2. **System action:** Fetch the database entries associated with various keywords’ related words, and previously accepted keywords, create a report to make the information readable and send the report to the user’s email address in a PDF file.

**UC-06:** User saving searches using the saved search option

The system allows the user to save searches in the database to increase the accuracy of keyword suggestions. The user can select the option to manually save a search through the home screen, and after submitting the combination, it will be added to the server under that user’s profile.

1. **User action**: User can save their searches using the save search option
2. **System action**: Create an entry in the database including the user’s specified word and related tags. The system shall return a message to confirm the entry was successfully added.

**UC-07:** User editing account details

The user can edit their account details, including password, name, age, email address, about section, and profile picture. For passwords, they need to put in their current password and put their new password into 2 fields.

1. **User action**: User changes their information in the fields of the web page.
2. **System action**: Details from the user’s entry in the database will be changed to the new data that the user is providing.

**6.0 Data Models/Entity-Relationship Diagram**

## 

Image 02: ER Diagram

**7.0 Traceability Matrix**

| **Traceability Matrix** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC-01 | UC-02 | UC-03 | UC-04 | UC-05 | UC-06 | UC-07 |
| FR-01 | x |  |  |  |  |  | x |
|  |  |  |  |  |  |  |  |
| FR-02 |  | x |  |  |  |  |  |
| FR-03 |  |  |  | x |  |  |  |
| FR-04 |  |  |  | x |  |  |  |
| FR-05 |  |  |  |  | x |  |  |
| FR-06 |  |  |  |  |  | x |  |
| FR-07 |  |  | x |  |  |  |  |

Table 01: Traceability Matrix

* FR-01 → UC-01
* FR-01 → UC-07
* FR-02 → UC-02
* FR-03 → UC-04
* FR-04 → UC-04
* FR-05 → UC-05
* FR-06 → UC-06
* FR-07 → UC-03

## 

## Appendix 4

### SDD - Software Design Document

**1. Architecture Design**

**1.1 Hardware Design**

The SiteSearchCLUE system shall be based upon a client/server architecture. Within this design, the server shall manage requests from the clients and retrieve the corresponding data from the database that stores the website and search information/indexes. Client machines shall communicate their requests for key search data and shall be processed by the server as they are received. This signifies a one-to-many relationship with multiple clients having the ability to communicate requests for resources with the server at any given time. The diagram presented in Figure 1.1 better illustrates this order of data flow.

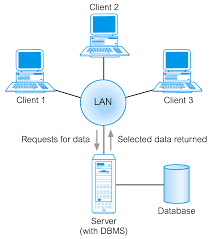


Figure 1.1: Client/Server Hardware Architecture

**1.2 Software Design**

The system is designed around a usable and intuitive User Interface. It shall easily allow users to perform their desired actions within the system using an understandable layout while also having easy access to any needed information.

The system is web-based. The web server will contain and handle most of the functional requirements and use-cases of the system. All user actions are processed on the webserver, including logging in, performing searches, and viewing saved search histories. During searches, the database server will be queried to search for and retrieve keywords for recommendations, which will then be relayed back to the user. In addition, during the login process and while requesting search histories, the database server will also be queried for the relevant information.

**2. Interface Design**

This Section specifies the look and feel of the search engine that will be visible to the user.

The following table identifies the pages and classes associated with each use case.

| **Use** | **Web Page / Class** |
| --- | --- |
| Sign Up | signup.html |
| Login/Logout | log.html |
| Saved Search | save.html |
| Search History | search.html |
| Manage Account | account.php |
| Personalized | personalized.html |
| Non-personalized | nonpersonalized.html |
| Settings | settings.php |

Table 2.1: Mapping Between Use-Cases and Web Pages / Classes.

## 

## 

**2.1 User Interface Specifications**

Following are the User interface specifications for the functionality listed in the above table.

**UI-01: UC-01&07:** User registering a new account in the system and editing it.

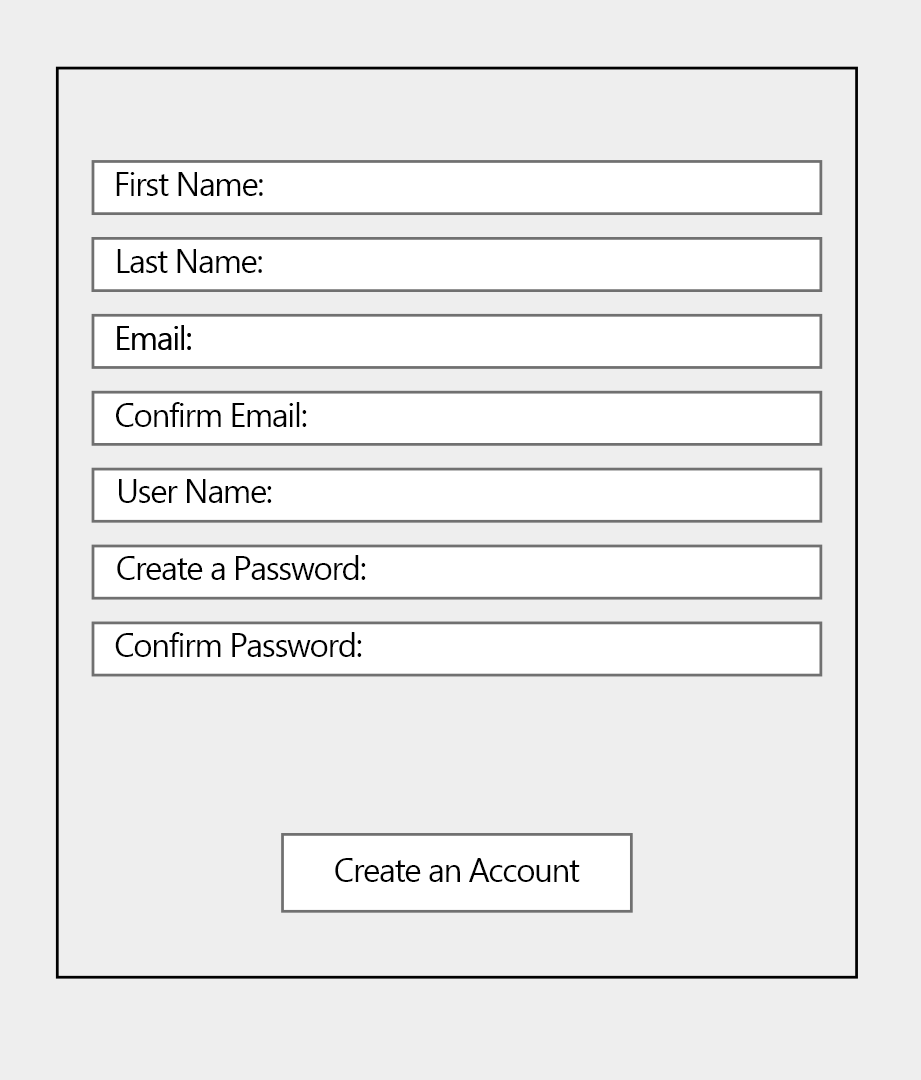


Figure 2.1: User signing up for an account.

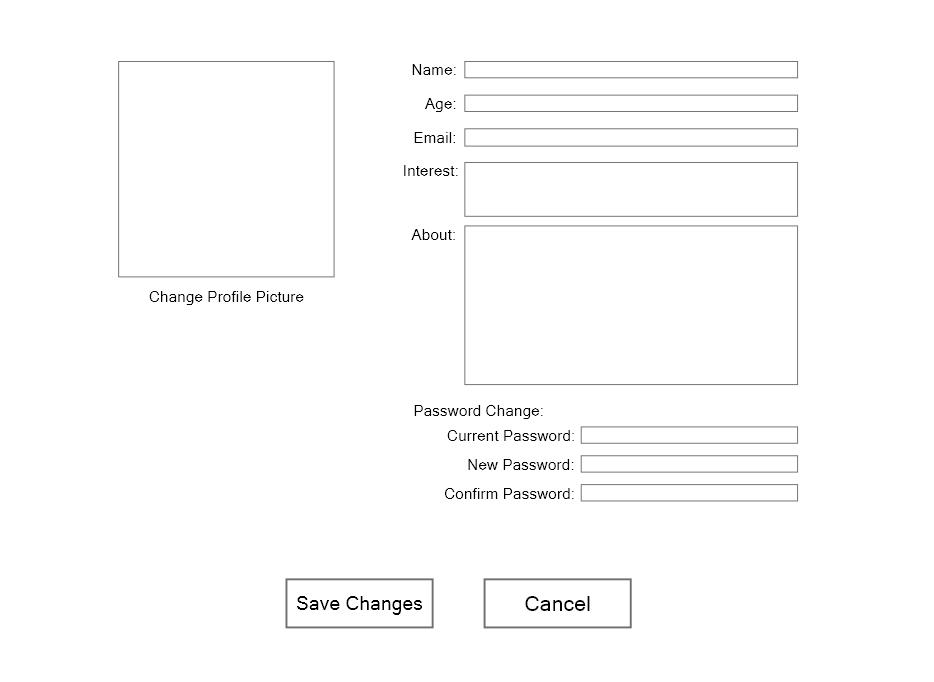


Figure 2.2: User can update their information

**UI-02: UC-02:** User logging into the system

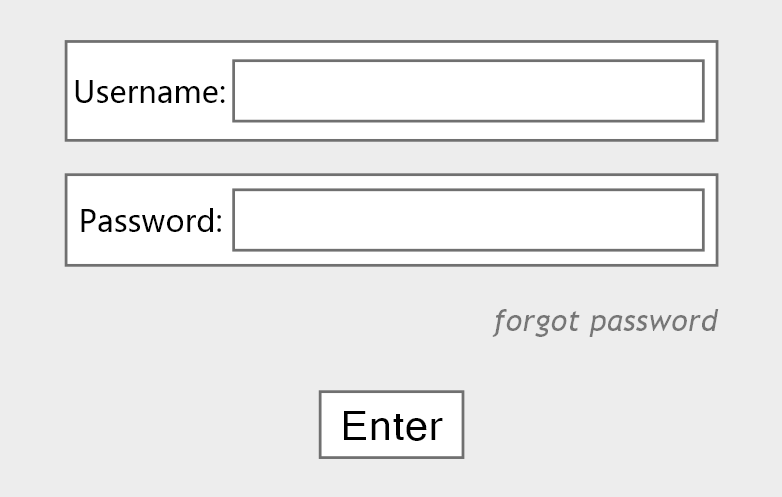


Figure 2.3: User login screen

**UI-03: UC-03:** User logging out of the system

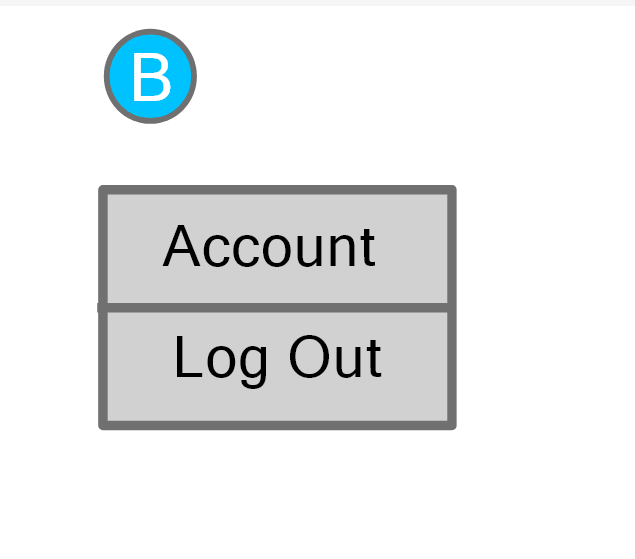


Figure 2.4: User log out option

**UI-04: UC-04:** User searching the website using string

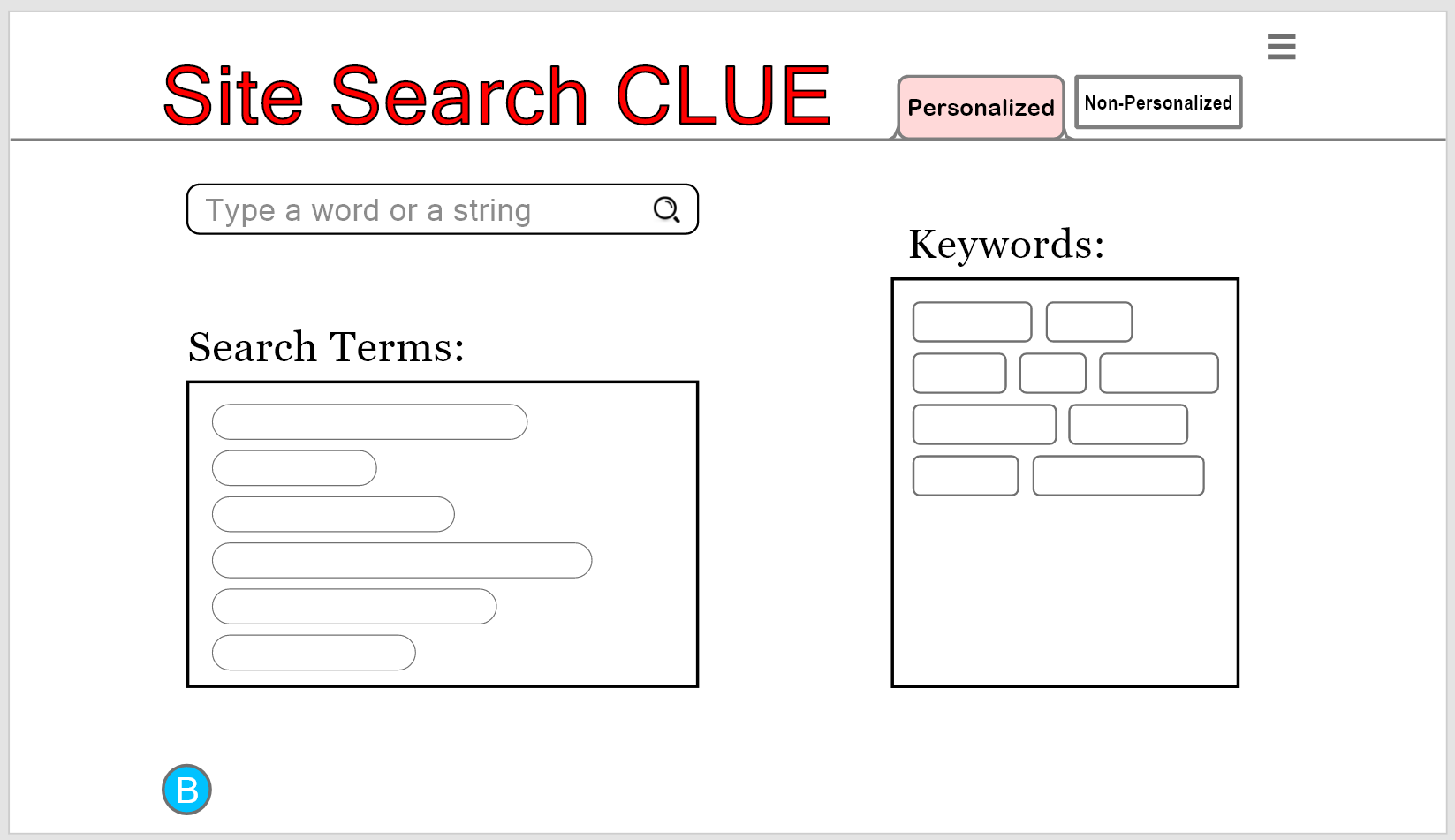


Figure 2.5: Personalized searching page

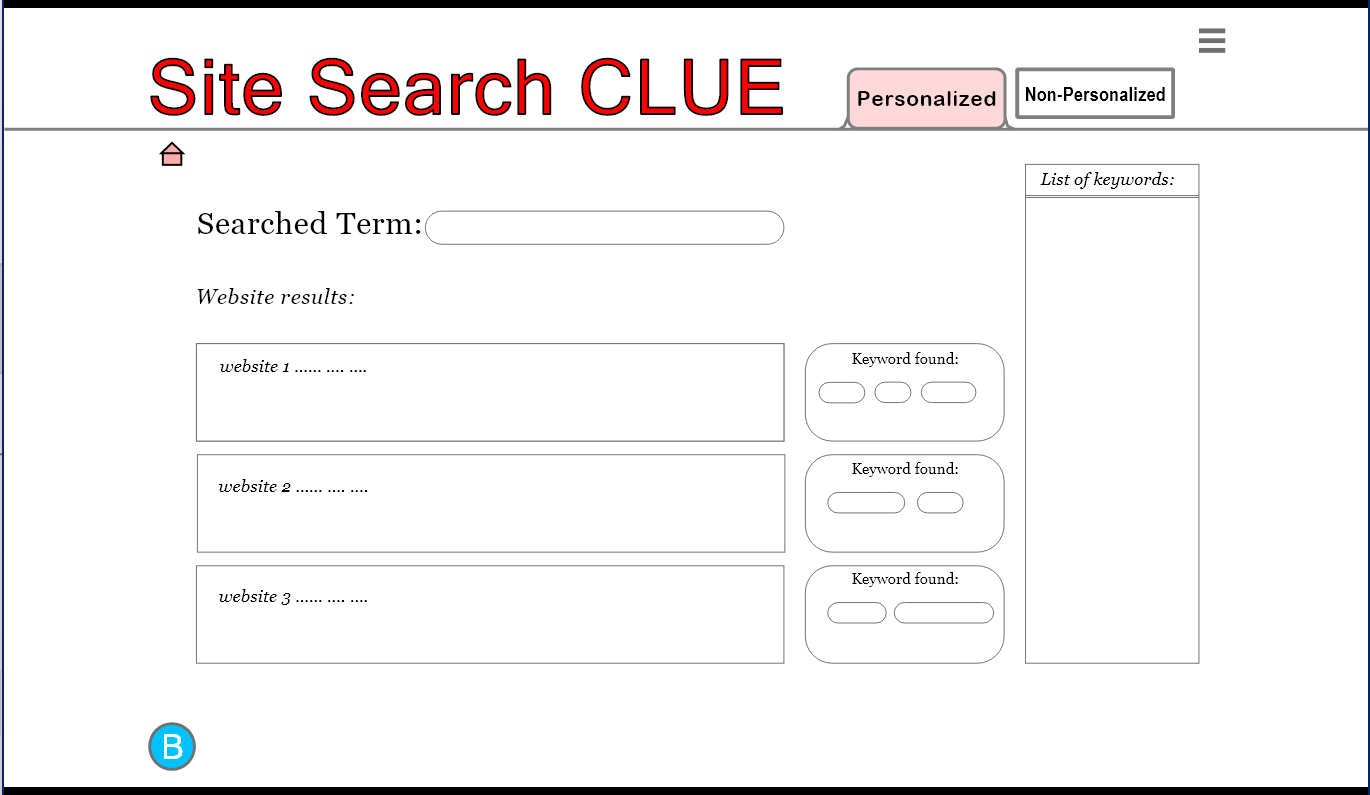


Figure 2.6: Personalized searching page - String Search

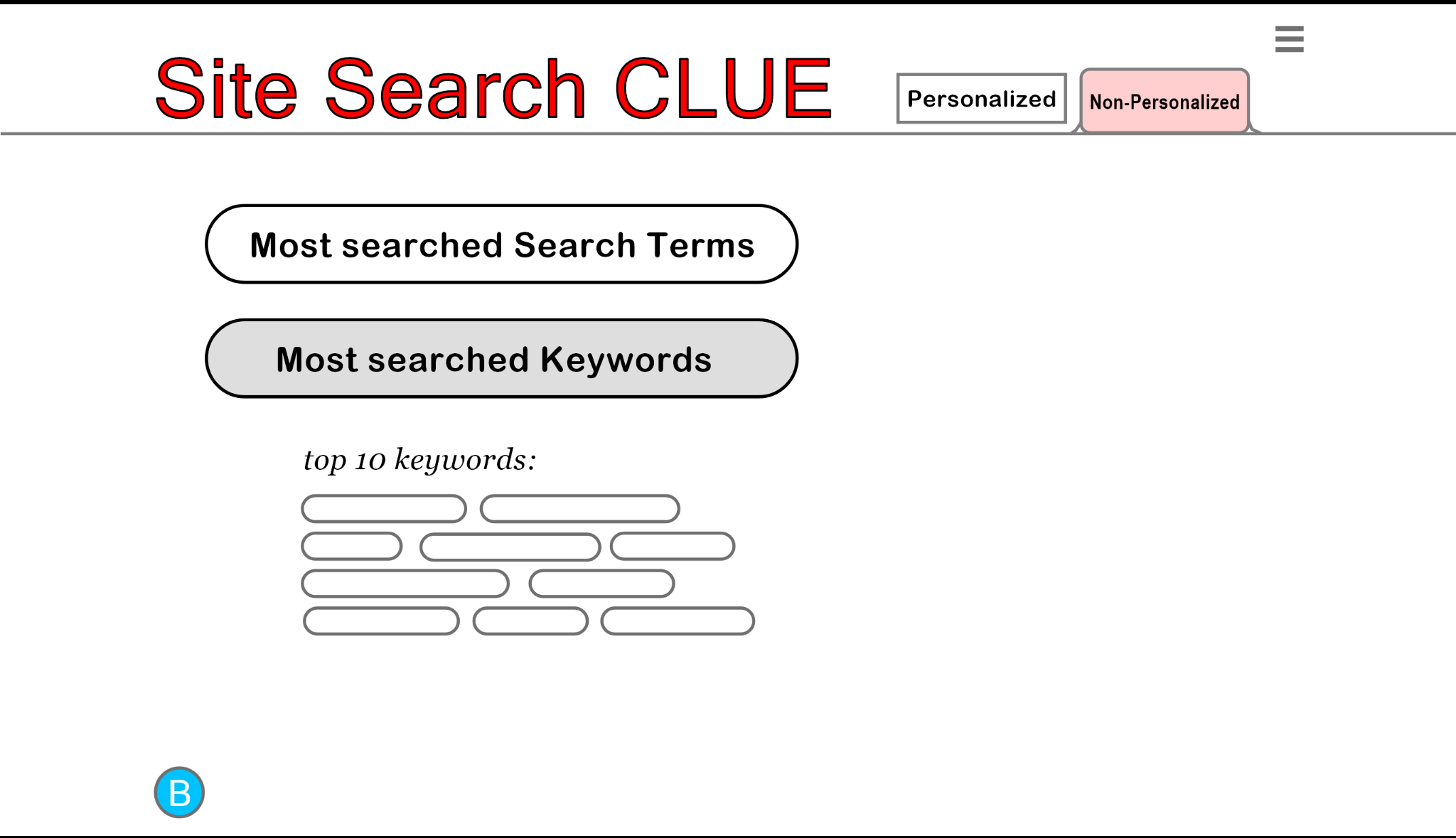


Figure 2.7: Non-personalized searching page

**UI-05: UC-05&06:** User accessing a history of searches and user saving searches using bookmarks and tags

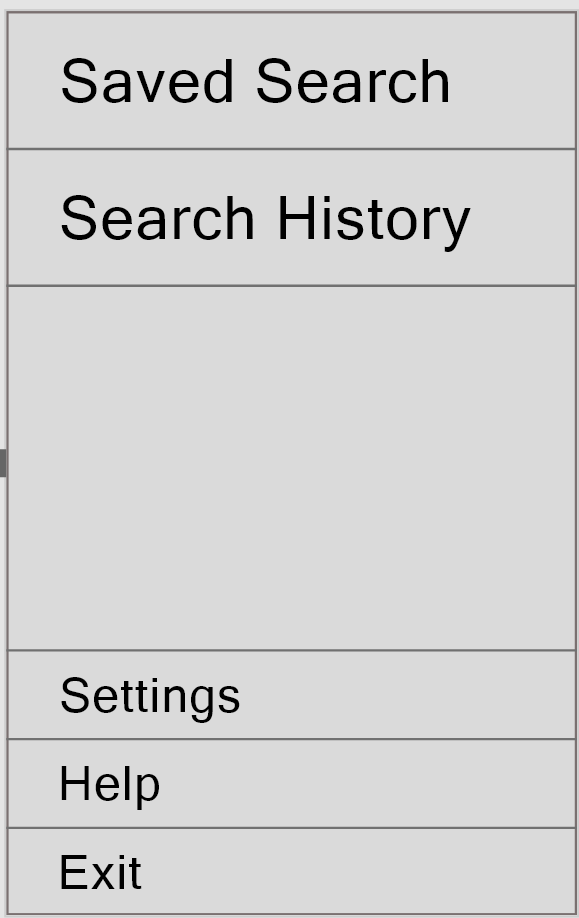


Figure 2.8: Bookmarks, Tags, Saved Searches and Search History options.

**3. Database Design**

**3.1 ER Design**

## 

In the **Users** table each entry describes a User.

In the **Websites** table, each entry describes a site on this current website that this search is being used in. The idea is that each unique domain has its database. If we decide to change this, we’ll need to make a “domains” table, and add a “domain” entry to each Websites entry.

In the **Site\_Keywords** table, each entry describes a keyword, what site it’s for, and who decided that that site should be tagged with that tag.

In the **Site\_History** table, each entry describes a website that a user accessed, and how many times that user accessed that website.

The **Search\_History** table is very similar to **Site\_History**, but instead, each entry is about a search term a user has searched, how many times the user has searched that, and if that search is saved for later.

**3.2 Creating the Database**

The following SQL statements will create the tables and relationships depicted on the prior page.

**#Parent Tables**

CREATE TABLE `Users`

(

`username` varchar(20) NOT NULL,

`password` char(64) NOT NULL,

`email` varchar(100) NOT NULL DEFAULT "",

`fname` varchar(100) NOT NULL DEFAULT "",

`lname` varchar(100) NOT NULL DEFAULT "",

`about` varchar(1000) NOT NULL DEFAULT "",

`age` tinyint(3),

PRIMARY KEY (`username`)

);

CREATE TABLE `Websites`

(

`URL` varchar(200) NOT NULL,

`title` varchar(50) NOT NULL DEFAULT "",

`text\_body` varchar(150) NOT NULL DEFAULT "",

PRIMARY KEY (`URL`)

);

**# Child Tables**

CREATE TABLE `Search\_History`

(

`username` char(20) NOT NULL,

`search\_string` varchar(200) NOT NULL,

`times\_accessed` int(7) NOT NULL DEFAULT 1,

`is\_saved` boolean NOT NULL DEFAULT FALSE,

PRIMARY KEY (`username`, `search\_string`),

FOREIGN KEY (`username`) REFERENCES `Users`(`username`)

);

CREATE TABLE `Site\_History`

(

`username` char(20) NOT NULL,

`URL` varchar(200) NOT NULL,

`times\_accessed` int(7) NOT NULL DEFAULT 1,

PRIMARY KEY (`username`, `URL`),

FOREIGN KEY (`username`) REFERENCES `Users`(`username`),

FOREIGN KEY (`URL`) REFERENCES `Websites`(`URL`)

);

CREATE TABLE `Site\_Keywords`

(

`URL` varchar(200) NOT NULL,

`keyword` varchar(50) NOT NULL,

`tagged\_by` varchar(20) NOT NULL DEFAULT "Admin",

PRIMARY KEY (`URL`, `keyword`),

FOREIGN KEY (`URL`) REFERENCES `Websites`(`URL`),

FOREIGN KEY (`tagged\_by`) REFERENCES `Users`(`username`)

);

**4. Component Design**

4.1 Component Design Specification

**Component 1: Register**

New user registers in the system.

**Precondition(s)**: The user must be a new user in the system

**Interface Specifications:** Contains 7 textboxes First Name, Last Name, Email, Confirm Email, User Name, Password, and Confirm Password. One button Create Account.

**Processing Specifications:** The user enters information in the required fields and then clicks the Create an Account button. The system shall then record this login name and password combination, and inform the user that the registration has been successful.

If (User supplies login name that does not already exist in the system)

Then (accept the user's registration attempt and inform the user that the

registration has been successful)

Else (present the user with a notification message alerting them that the login name already exists.)

**Database Requirements:** Users must be in the system for this use case to work.

**Postcondition(s):** Access to signup.html

**Component 2: Login**

User logging into the system.

**Precondition(s)**: The user must be a valid user in the system

**Interface Specifications:** Contains 2 textboxes Name and Password. Two buttons,

Enter and forgot password.

**Processing Specifications:** The user enters information in the required fields and then clicks the Enter button. The required fields are then tested for authentication; if successfully verified then it will send the users to the main page according to their level of authority. If the user does not remember their password they click the forgot password screen and are then redirected.

If (Username and password are correct)

Then (Send user authentication token back)

Else (Send error)

If (received user authentication token)

Then (continue to search.html)

**Database Requirements:** The user’s credentials must be in the database for this use case to work.

**Postcondition(s):** Access to search.html.

**Component 3: Logout**

User logging out of the system.

**Precondition(s)**: The user must be logged into the system

**Interface Specifications:** Two buttons, Account and Log Out.

**Processing Specifications:** The user clicks the Log Out button. If valid the user is then returned to the main screen.

If (user is logged in)

Then (log user out)

**Database Requirements:** None

**Postcondition(s):** Access to log.html.

**Component 4: History**

User accessing a history of searches and saving searches.

**Precondition(s)**: The user must have logged into the system.

**Interface Specifications:** Contains 5 textboxes: Saved Search, Search History, Settings, Help, Exit.

**Processing Specifications:** The user clicks on the option they want to access. If they click Saved Search, Search History, and Setting. They take you to their respective menus.

If (The user supplies valid attribute values for each possible attribute of the new record,

and the ID attribute value does not equal the ID attribute value for an existing record)

Then (accept the user's insertion attempt and create a new record in the database )

Else If (User fails to supply values for all of the possible attributes of a record in the database)

Then (inform the user that to insert a new record into the database, each

attribute's input fields must contain a value)

Else(reject the user's insertion attempt and notify the user that the ID attribute value

corresponds to an existing record)

**Database Requirements:** DB stores the site viewing history of each user, and separately saves sites that users have decided to save.

**Postcondition(s):** Access to search.html.

**Component 5: Manage Account**

User changing their account details

**Precondition(s)**: User must have logged into the system.

**Interface Specifications:**

**Processing Specifications:** The user types into the fields they wish to save. If they click on the “change profile picture” text box, then an image prompt will show up for them to upload an image from their computer to be their profile picture. After doing so, they can click the “save changes” text box to save any changes they have made to the information. If the current password field doesn’t match the user’s current password, or the confirm password field doesn’t match the new password field, then when the “Save Changes” text box is clicked all changes are saved except for the new password, which is discarded and the old password is kept.

If (User uploads image )

Then (Replace current profile picture)

If (User changes password)

Then (Discard old password and set a new password)

If (User selects save changes)

Then (Actualize all changes)

Else If (User does not select save changes)

Then (Discard all changes)

**Database Requirements:** The database stores the information from the user in the database. It converts the profile picture to .jpg, renames it to [User’s Username Here].jpg, and stores it in a directory titled “ProfilePictures”.

**Postcondition(s):** Access to account.php

**Component 6 Personalized:**

User using their personalized searches to search for websites

**Precondition(s)**: The user must have logged into the system.

**Interface Specifications:** Search option, Search terms, and Keywords

**Processing Specifications:** Users can select a search term from the personalized search terms or keywords that are already given to the user or lookup web pages using their preferred search string.

If(User has any recorded searches)

Then (provide personalized search terms)

Else

Then(only provide non-personalized)

**Database Requirements:** The database must contain different users’ search histories for this use case to work.

**Postcondition(s):** Access to personalized.html

**Component 7 Non-Personalized:**

User using their non-personalized searches to search for websites

**Precondition(s)**: User must be logged into the system.

**Interface Specifications:** Most searched Search Terms and Keywords

**Processing Specifications:** Users can select the search terms from the most used search terms and keywords to look for websites of their choice.

If(User selected search terms)

Then (Provide appropriate list in correspondence with the filter)

**Database Requirements:** Users must be in the system for this use case to work.

**Postcondition(s):** Access to nonpersonalized.html

**Component 8 Settings:**

Users can use settings to customize their preferences

**Precondition(s)**: The user must have logged into the system.

**Interface Specifications:** Settings option

**Processing Specifications:** Users can use the settings options to customize their preferences and their choices, they can select which category they might want to search.

If(User provides changes to features)

Then (finalize changes to the user's account)

**Database Requirements:** Users must be in the system for this use case to work.

**Postcondition(s):** Access to settings.php

**5. Software Requirements Traceability Matrix**

| FR | UC | Use case Description | Component | UI |
| --- | --- | --- | --- | --- |
| **FR-01** | **UC-01** | User registering their account | **COM-01** | **UI-01** |
| **FR-01** | **UC-07** | User editing account details | **COM-05** | **UI-01** |
| **FR-02** | **UC-02** | User logging in | **COM-02** | **UI-02** |
| **FR-03** | **UC-04** | User searching | **COM-06&07** | **UI-04** |
| **FR-04** | **UC-04** | User search stored to compare | **COM-04** | **UI-04** |
| **FR-05** | **UC-05** | User accessing their history of searches | **COM-04** | **UI-05** |
| **FR-06** | **UC-06** | User saving searches | **COM-04** | **UI-05** |
| **FR-07** | **UC-03** | User logging out | **COM-03** | **UI-03** |

### 

## 

## Appendix 5

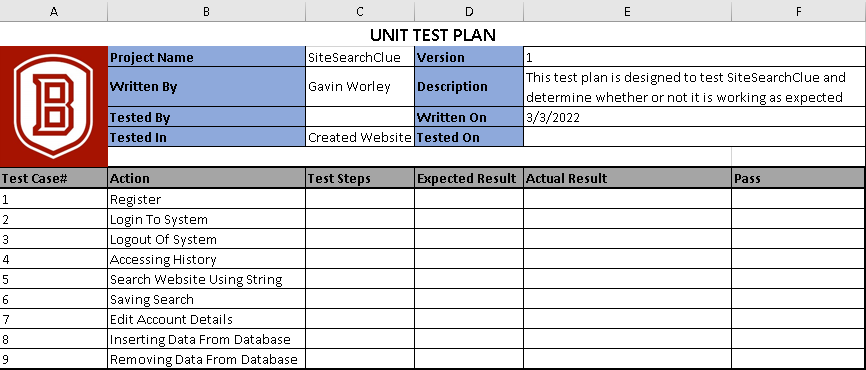
### STD (AT) - Software Testing Document (Application Testing)

**1.0 Test Plan**

1.1 Unit Test Plan

This section of testing will assure that the basic user functionality of the website is working. We implemented the functionality of unit testing into the Python API backend. Our testing is beginning at this low, “alpha” level before moving to the higher, “beta” level.

See the screenshot below for the unit test plan.



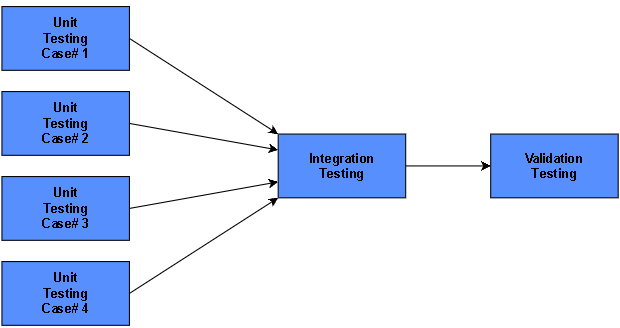
1.2 Integration Test Plan

This section of testing will ensure that the implementation of the system is without noticeable flaws. We’ve chosen a bottom-up approach to integration testing, that way we can nail down the basic functionality before looking at that larger system.

| **ID** | **Integration Test** | **Description** |
| --- | --- | --- |
| I1 | Database manager -> HTML page | Only registered users can log in to search things. |
| I2 | Database manager -> HTML page | Users are able to search. |
| I3 | Database manager -> HTML page | Only the database manager can insert into/delete from/update the database. |
| I4 | Database manager -> HTML page | Users are only able to change their own personal account details. |

1.3 Validation Test Plan

This section of testing assures that the software will meet all of the client’s requirements. Also, it demonstrates that the software will fulfill its intended purpose when deployed.



# 

**2.0 Test Cases**

2.1 Unit Test Cases

| **Test Case #** | **Test Cases** | **Input Test Data** | **Expected Output** |
| --- | --- | --- | --- |
| 1 | Register | Username, password, and email address | Successful registration (verifies there are no duplicate accounts and sends confirmation email). Adds the user to database records. |
| 2 | Log in to the system | Username or email address, and password | The user must be able to access their account. |
| 3 | Logout of system | Logout request | The user must be able to exit their account and return to normal browsing. |
| 4 | Searching website using string | String query within search field | Generate search suggestions based on keywords within the database and past user searches. |
| 5 | Accessing search history | User search history request | Returns database records of all past searches of the user. |
| 6 | Saving search | Save search request and string query | Create an entry in the database including the user’s search query and related tags. |
| 7 | Editing account details | New password, new username, or new email | User account details must be updated with no errors. |

## 

**2.2 Integration Test Cases**

Integration test case I1:

| **Test case Identifier** | I1 |
| --- | --- |
| **Test Item(s)** | HTML Page -> Database Manager |
| **Input specifications** | Username, password, and email address. |
| **Output**  **specifications** | Check if the Administrator is able to perform the below actions correctly:   1. Add new entry to the user database 2. Update database record with user credentials |
| **Environmental Needs** | Browser |

Integration test case I2:

| **Test case Identifier** | I2 |
| --- | --- |
| **Test Item(s)** | HTML Page -> Database Manager |
| **Input specifications** | Login details of the already registered user. |
| **Output**  **specifications** | Check if the user is able to log in without any errors. |
| **Environmental Needs** | Browser, I1 is successful. |

Integration test case I3:

| **Test case Identifier** | I3 |
| --- | --- |
| **Test Item(s)** | HTML Page -> Database Manager |
| **Input specifications** | Search query (in string format) |
| **Output**  **specifications** | Check if string query broadly matches any keywords within the database or past search database. |
| **Environmental Needs** | Browser, I2 is successful. |

**3.0 Test Procedure**

UC1: Register

**Flow of Events:**

**Basic Flow:**

*User Action:*

The user enters an email, password, and unique username.

*Response:*

The user’s input is used to create a new profile in the database that can be used later to log in.

**Alternative Flow 1:**

*User Action:*

The user enters a username that is already used by an existing user.

*Response:*

The account registration is rejected, and the user is asked to attempt to register with a different username.

| Scenario 1 | Basic Flow |
| --- | --- |
| Scenario 2 | Alternative Flow 1 |

| Test Case ID | Scenario | Condition | Expected Result |
| --- | --- | --- | --- |
| TC1 | Scenario 1 | Username = “a”, password = “a”, email = “sample@sample.net” | The user will successfully be able to create an account. |
| TC2 | Scenario 2 | Username = “a”, password = “b”, email = “sample@sample.com”, after the previous test | The user will be unable to successfully be able to create an account because an account with that username already exists. |

### 

UC2: Log in

**Flow of Events:**

**Basic Flow:**

*User Action:*

The user enters a Username and Password that exists in the system.

*Response:*

The user is logged into their account.

**Alternative Flow:**

*User Action:*

The user enters a Username and Password that does not exist in the system.

*Response:*

The user will be prompted to try and login again.

| Scenario 1 | Basic Flow |
| --- | --- |
| Scenario 2 | Alternative Flow 1 |

| Test Case ID | Scenario | Condition | Expected Result |
| --- | --- | --- | --- |
| TC3 | Scenario 1 | Username = “a”, password = “a” | This user exists and they will be logged in. |
| TC4 | Scenario 2 | Username = “a”, password = “b” | This user will be prompted to try and log in again. |

### 

UC3: Log out

**Flow of Events:**

**Basic Flow:**

*User Action:*

The user selects the logout option.

*Response:*

The user is logged out of their account.

**Alternative Flow:**

*User Action:*

The user waits 5 minutes without any activity.

*Response:*

The user is logged out of their account.

| Scenario 1 | Basic Flow |
| --- | --- |
| Scenario 2 | Alternative Flow 1 |

| Test Case ID | Scenario | Condition | Expected Result |
| --- | --- | --- | --- |
| TC5 | Scenario 1 | Log Out option | The user is logged out. |
| TC7 | Scenario 2 | Inactive for 5 minutes. | The user is logged out. |

### 

UC4: Search

**Flow of Events:**

**Basic Flow:**

*User Action:*

The user is logged in.

*Response:*

The user receives multiple personalized search term recommendations based on what similar users search.

*User Action:*

The user inputs a term into the search bar and hits the search button.

*Response:*

The system shows a list of the results starting with what is most related to the search term

**Alternative Flow:**

*User Action:*

The user is not logged in.

*Response:*

The user receives multiple non-personalized search term recommendations based on what is popular.

*User Action:*

The user inputs a term into the search bar and hits the search button.

*Response:*

The system shows a list of the results starting with what is most related to the search term.

| Scenario 1 | Basic Flow |
| --- | --- |
| Scenario 2 | Alternative Flow 1 |

| Test Case ID | Scenario | Condition | Expected Result |
| --- | --- | --- | --- |
| TC6 | Scenario 1 | User inputs search term | Results are shown in relation order. |
| TC8 | Scenario 1 | User is logged in | Able to view personalized search term recommendations |
| TC9 | Scenario 2 | User isn’t logged in | Able to view non-personalized search term recommendations |
| TC10 | Scenario 1 | User is logged in | Able to view non-personalized search term recommendations |

**4.0 Traceability Matrix**

| **FR** | **Use Case** | **UC Description** | **Component** | **UI** | **Test Cases** |
| --- | --- | --- | --- | --- | --- |
| FR-01 | UC-01 | User registering a new account in the system. | COM-01 | UI-01 | TC1, TC2 |
| FR-01 | UC-07 | User editing account details | COM-05 | UI-01 | TBD |
| FR-02 | UC-02 | User logging into the system | COM-02 | UI-02 | TC3, TC4 |
| FR-03 | UC-04 | User searching the website using string | COM-06&07 | UI-04 | TC6, TC8, TC9 |
| FR-04 | UC-04 | User search stored to compare with other users | COM-04 | UI-04 | TBD |
| FR-05 | UC-05 | User accessing a history of searches | COM-04 | UI-05 | TBD |
| FR-06 | UC-06 | User saving searches | COM-04 | UI-05 | TBD |
| FR-07 | UC-03 | User logging out of the system | COM-03 | UI-03 | TC5 |

\*\*We decided to prioritize our use cases in order of 1, 2, 3, 4, 5, 6, 7. We plan to create more test cases over the course of the next 2 weeks and over the course of later iterations

## 

## Appendix 6

### ATR - Accepting Test Report

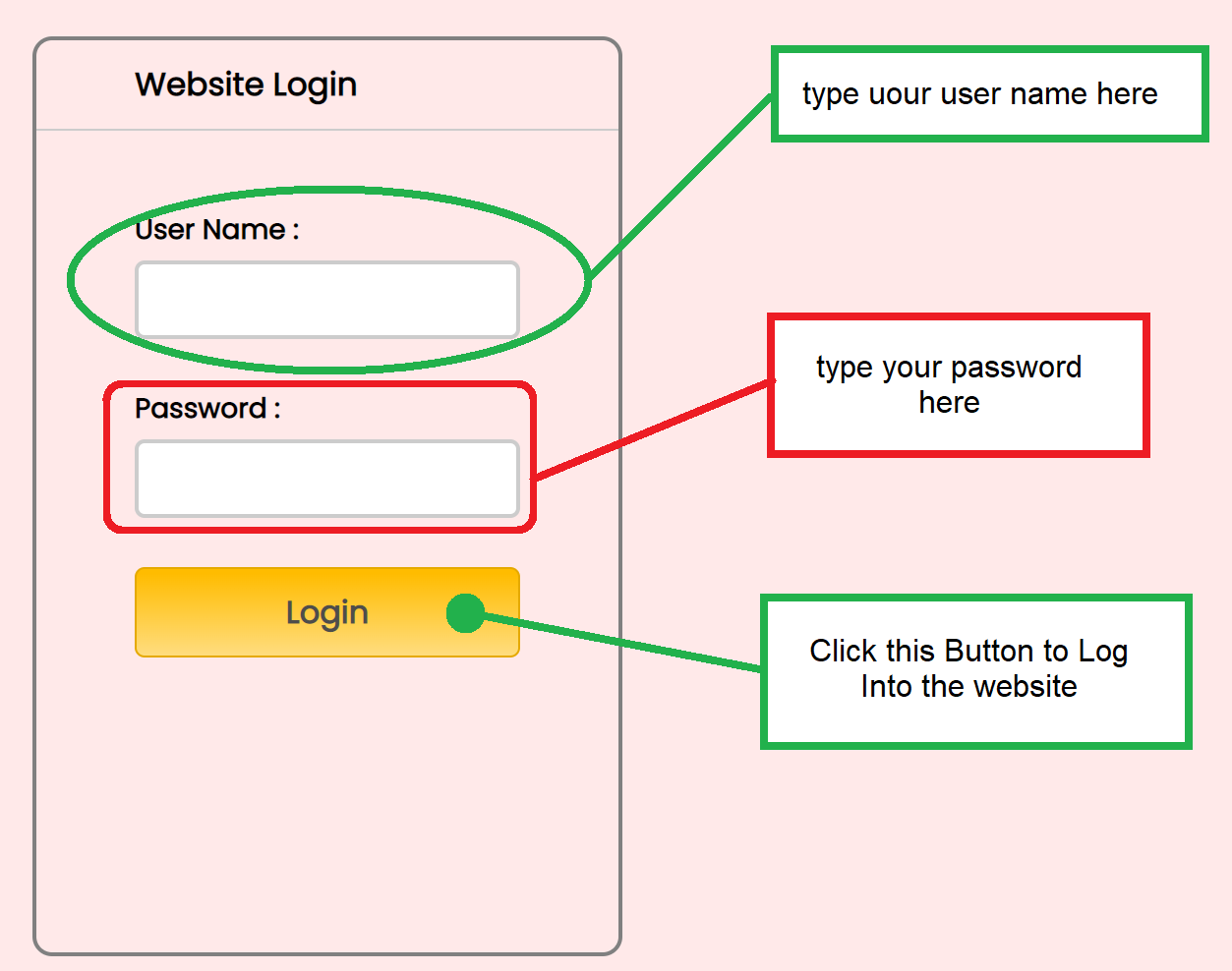
| **Test Case#** | **Test Result** | **Requirements** | **Comments** |
| --- | --- | --- | --- |
| 1 | Fail | Register Account | Work in Progress |
| 2 | Fail | Unable to create an account with identical username | Work in Progress |
| 3 | Pass | Able to log into the system | The system confirms when login is successful. |
| 4 | Pass | Not able to log into the system with incorrect credentials | Users are given 3 attempts to log in. |
| 5 | Pass | Able to log out | Successfully log the user out. |
| 6 | Pass | Search results appear after searching. | Results show up |
| 7 | Fail | Logged out after 5 minutes of inactivity. | Work in Progress |
| 8 | Pass | User is recommended Personalized search terms when logged in | User is recommended 5 personalized search terms |
| 9 | Pass | User is recommended Nonpersonalized search terms when logged out | User is recommended 3 non-personalized search terms |
| 10 | Pass | User is recommended Nonpersonalized search terms when logged in | User is recommended 3 non-personalized search terms |

## Appendix 7

### User Manual

**Log In**

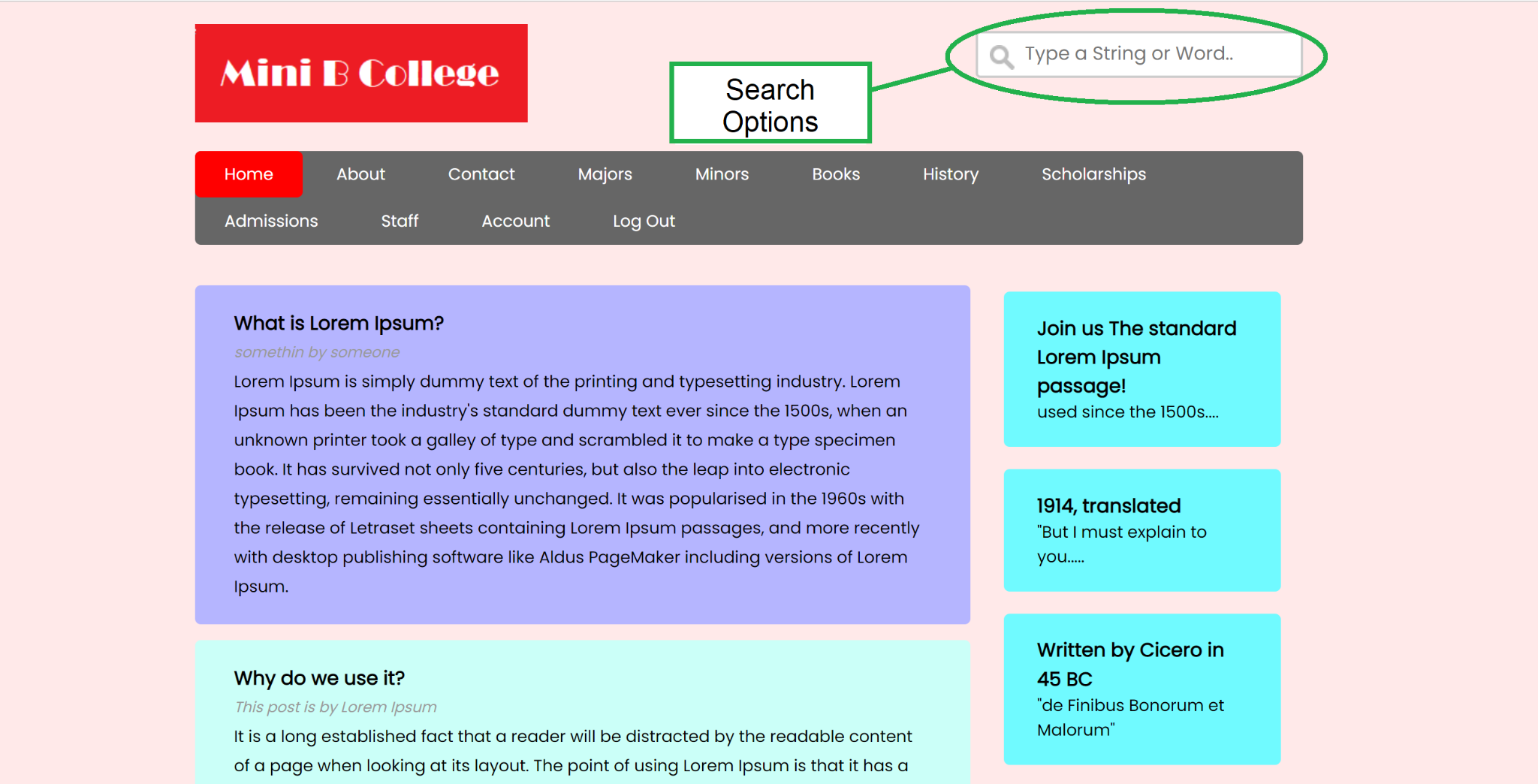
You can log in to the website from the main page by filling in your username and password and clicking on the button underneath.

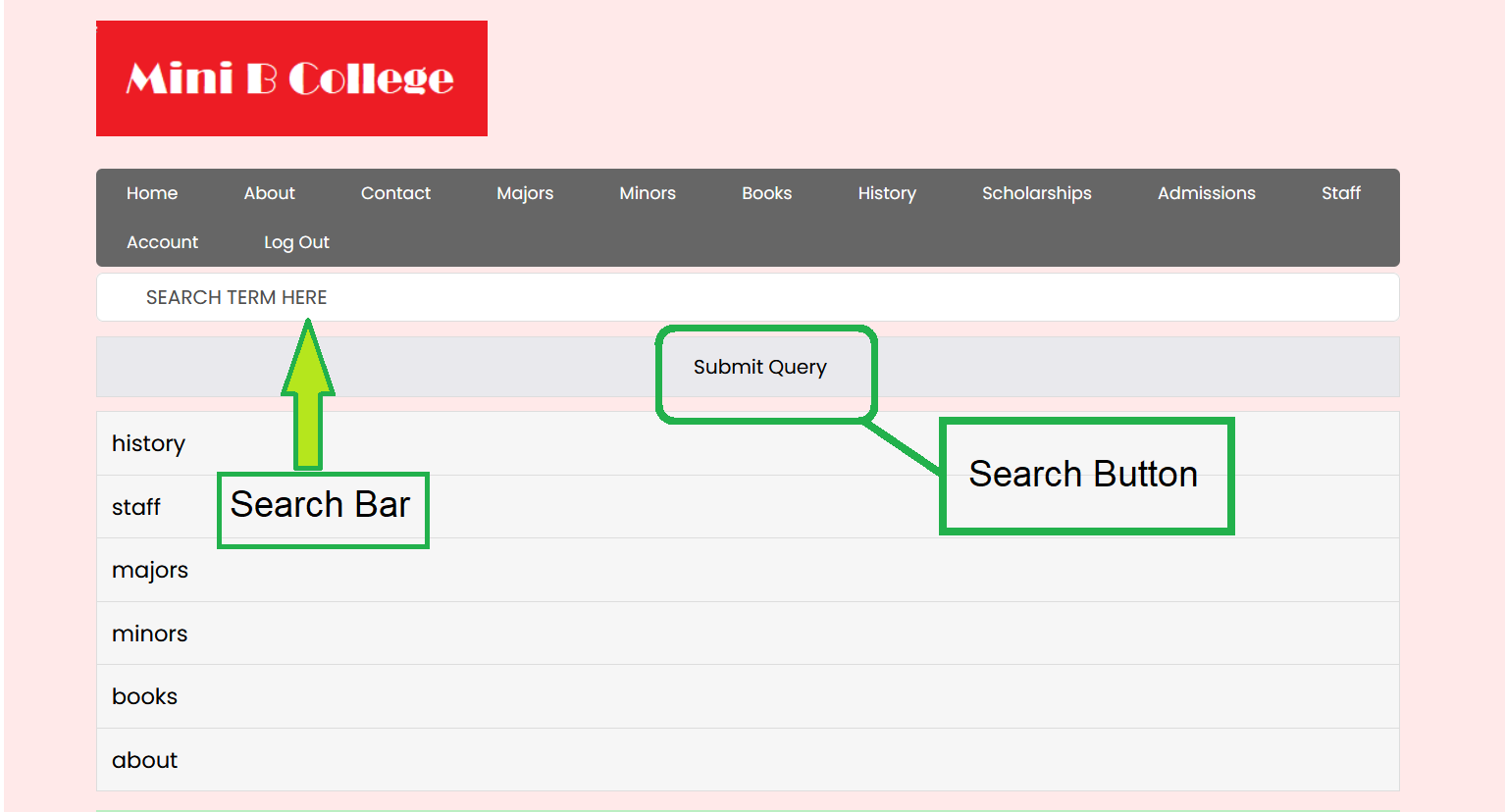


# 

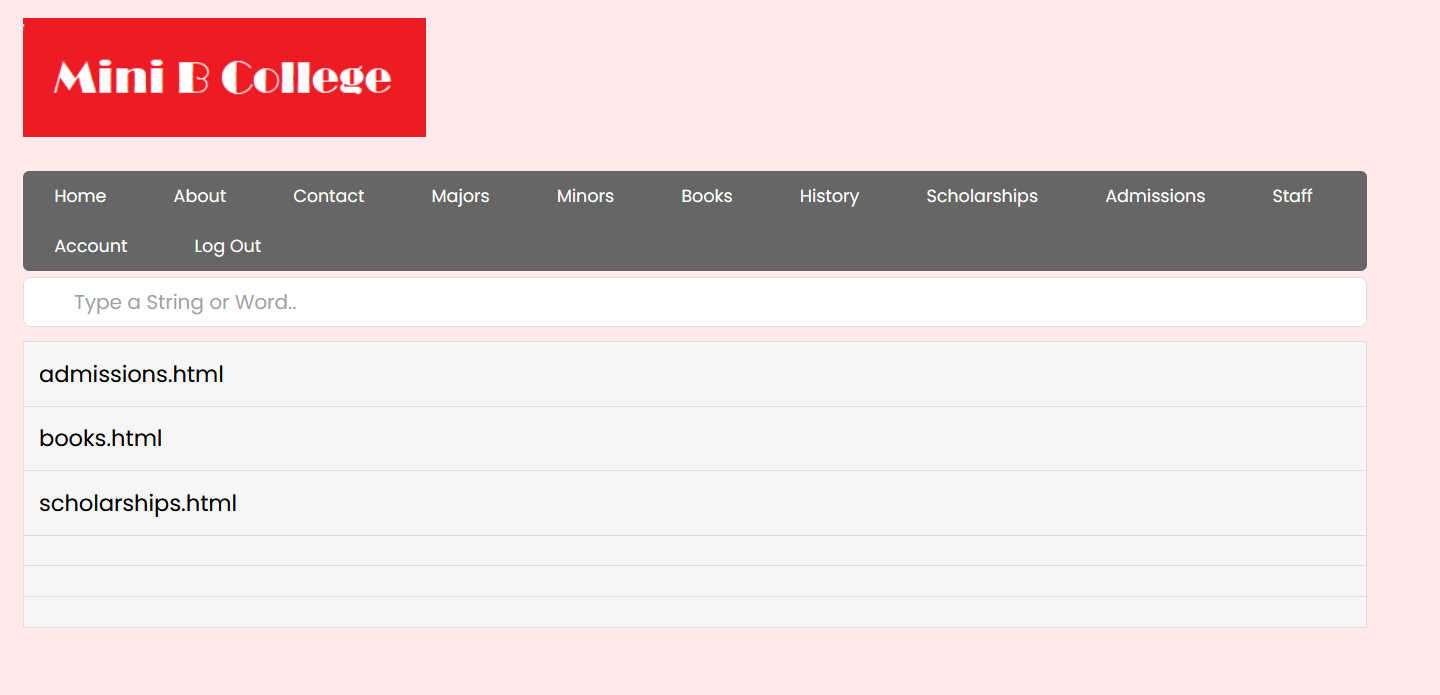
**Search**

You can search the website by clicking on the search bar on the top right after logging in. You will be taken to the /search page, then shown a list of URLs for your search.



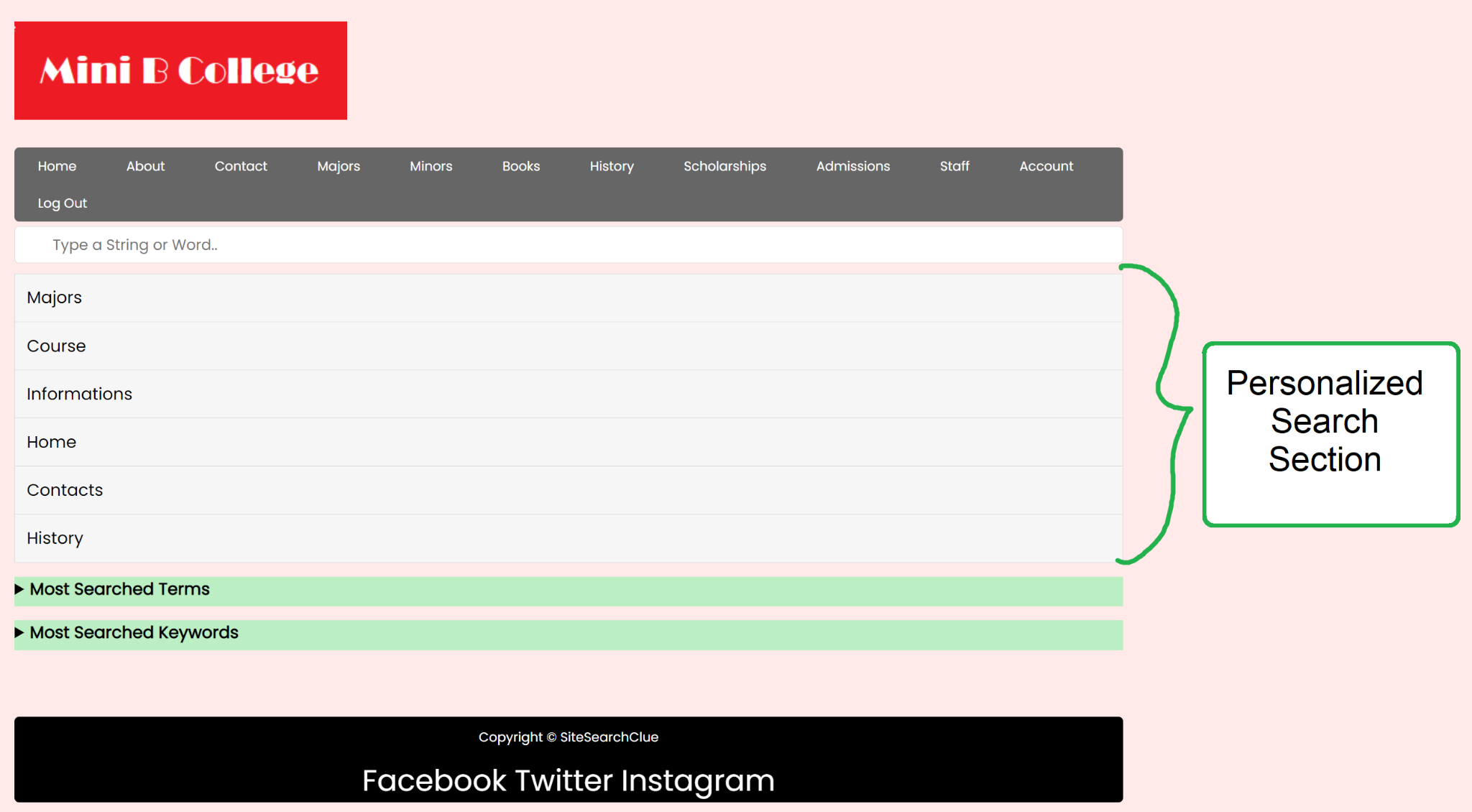


Below are the results for searching “financial”:



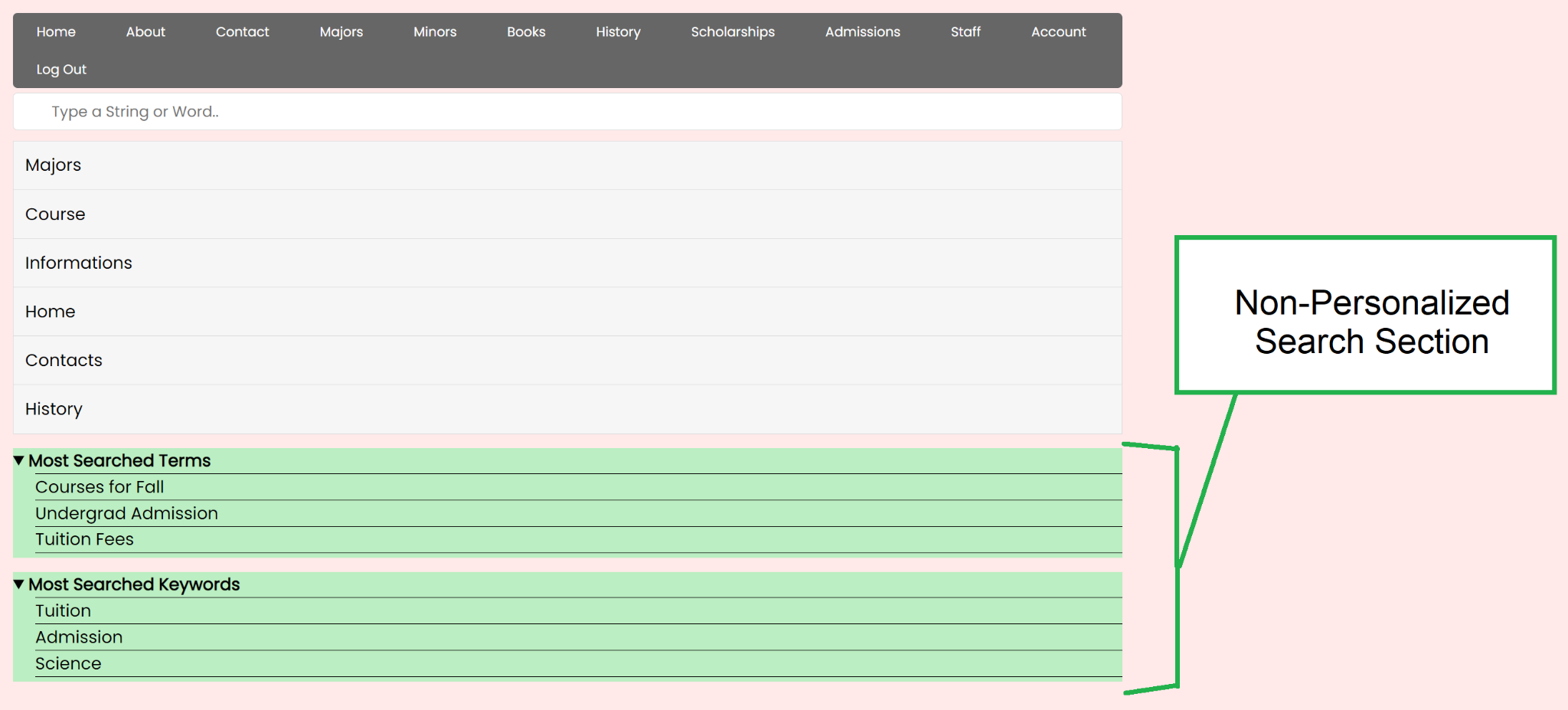
**Personalized Recommendations**

Personalized recommendations are recommendations based on similar users. Cosine similarity is used to determine how similar each user is, this allows the system to base recommendations on what the other users would “recommend”. Your personalized recommendation results are displayed underneath the search box.

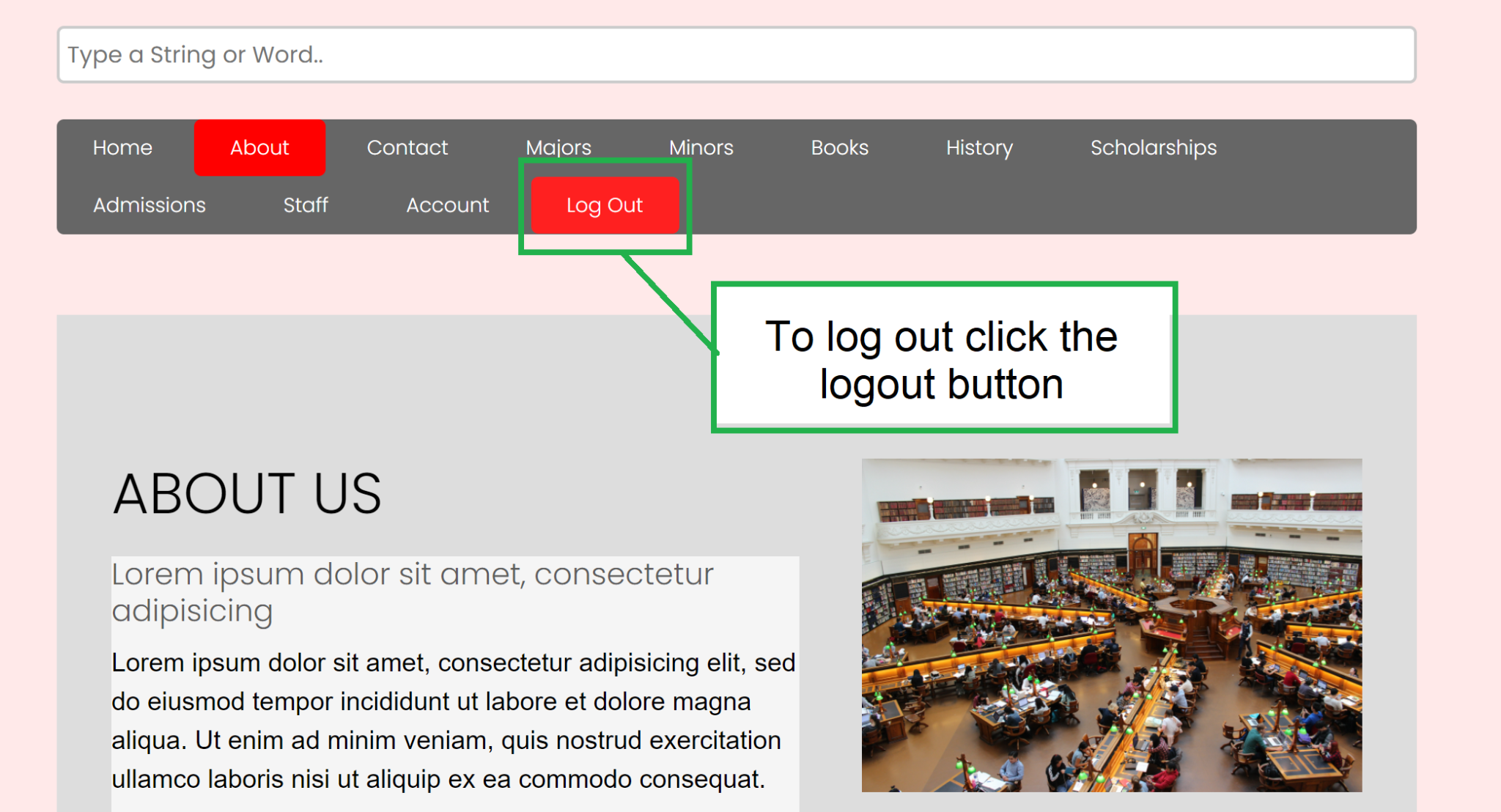


**Non-Personalized Recommendations**

Non-personalized recommendations are recommendations gathered from the website as a whole that do not require the user to be signed in. Non-personalized recommendations will be shown as a list of the most used search terms.



**Log Out**

When you log out, you will be taken back to the /login.html page. If you are logged out, you will no longer be able to search, however, you will still see non-personalized recommendations and be able to view them.

## Appendix 8

### Expo Abstract

When searching a website for information, choosing the appropriate search terms is critical to a successful search. However, it may be challenging, especially for the inexperienced. Thus, we’re developing SiteSearchCLUE: a search-term recommender system that provides both non-personalized and personalized search terms within a website during the website search process. It’s currently running on a sample university website and is developed using Python-Surprise.

The purpose of our project is to assist visitors with their website searches, whether they are looking for something specific or just browsing, SiteSearchCLUE will customize and tailor terms that may be helpful to specific visitors and assist them whether precision-oriented or recall-oriented searches, choosing frequent search phrases.

SiteSearchCLUE provides both non-personalized and personalized clues for search phrases based on a visitor's profile:

* *Non-personalized*:

1. Search terms that visitors have entered most into the website search box & Keywords describing the web pages that visitors have visited most at a website and
2. All available keywords describe the web pages that are retrieved at a given search step.

* *Personalized*: Search terms that similar visitors have entered into the website search box & Keywords describing the web pages that similar visitors have visited using:

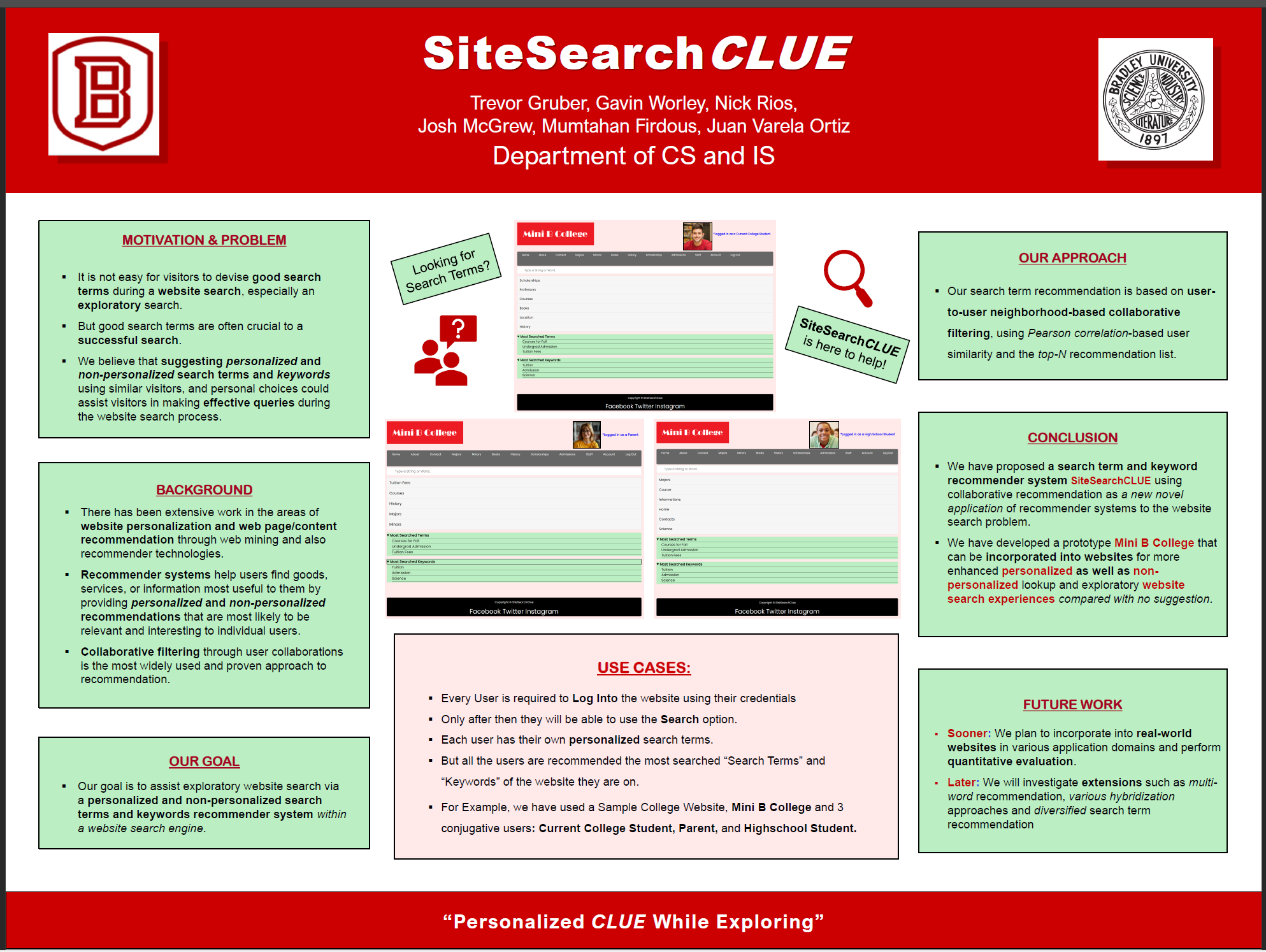
1. The history of all visitors’ search terms and
2. The history of all visitors’ web pages via user-to-user collaborative filtering.

By recommending non-personalized and personalized words as prospective search keywords, SiteSearchCLUE assists website users in browsing and selecting better search terms.

## 

## Appendix 9

### Expo Poster



## 

## Appendix 10

### Demo Video

[](https://drive.google.com/file/d/1R3BsY_opSpXgLZSPDCr6AGDXqL81M7L_/view?usp=sharing)

# **Iteration Reports**

## Iteration-1

Developing SRS:

**Part-1**: (10.13.21-10.22.21) We started by distributing the work among us. And throughout the week, we had 5 meetings and discussed our progress and what needs to be done and updated.

**Part-2:** (10.23.21-11.04.21) We had 4 meetings in the last 2 weeks after the 1st iteration period and discussed what needed to be updated and updated accordingly.

Here is a table for work distribution and updates:

| **Task** | **Date Assigned** | **Initially Assigned Person** | **Status** | **Date**  **Finished** |
| --- | --- | --- | --- | --- |
| 1.0 Stakeholders Identification | 10/13/21 | Nicolas Rios | ***Updated*** | 11/04/2021 |
| 2.0 Functional Requirements | 10/13/21 | Mumtahan Firdous | Completed | 10/20/2021 |
| 3.0 Nonfunctional Requirements | 10/13/21 | Nicolas Rios | ***Updated*** by other group members | 11/03/2021 |
| 4.0 Other Constraints | 10/13/21 | Josh McGrew | Completed | 10/20/2021 |
| 5.0 Use Case Model   | 5.1 Actors | 10/13/21 | Gavin Worley | Completed | 10/20/2021 | | --- | --- | --- | --- | --- | | 5.2 Use-Cases | 10/13/21 | Gavin Worley | Completed | 10/20/2021 | | 5.3 Use-Case Diagram | 10/13/21 | Nick Stuart | Completed | 10/20/2021 | | 5.4 Use-Case Descriptions | 10/13/21 | Nick Stuart | ***Updated*** | 11/2/2021 | | | | | |
| 6.0 Data Models/  Entity Relationship Diagram | 10/13/21 | Trevor Gruber | ***Updated*** | 10/31/2021 |
| 7.0 Traceability Matrix | 10/13/21 | Juan Varela Ortiz | Completed | 10/20/2021 |

Practicing Development Tools: (10/13-11/04)

Gavin: I practiced a bit of Python and looked into the libraries NumPy and PyTorch. I also started looking a bit into HTML.

Josh: Used Python as well as the Pandas and Numpy libraries. Reviewed HTML and SQL.

Juan: Python, HTML as well as SQL.

Mumtahan: As I took the section to work on the UX and UI design, I have been looking into the tools. So far I got to practice Adobe XD, but I still need to get more into that and some other tools. Looked through the libraries, but haven’t made much progress yet. Started looking into Prototypes for next week.

Nick R: Have reviewed Javascript Bootstrap and other jQuery plug-ins for usage in the front-end development of the software.

Nick S: Reviewed software engineering principles to create the use case diagram. Also read about the Python libraries to get more familiar with them.

Trevor: Reviewed basic Python syntax, keywords, etc. Also, I refreshed myself on database fundamentals and MySQL data types.

## Iteration-2

| **Task** | **Assigned date** | **Current status** | **Date Finished** | **Person** |
| --- | --- | --- | --- | --- |
| Iteration-2 | 11/6/21 | Done | 12/1/2021 | Group Work |
| **Software Prototype**   | Sample Website | 11/6/21 | In Progress | **--/--/----** | Nick Stuart | | --- | --- | --- | --- | --- | | Prototype | 11/6/21 | Done | 12/1/2021 | Mumtahan Firdous | | | | | |
| **SDD (**Software Design Document**)**   | 1. Architectural Design   | 1.1 Hardware Design | 11/6/21 | Done | 11/17/2021 | Nick Rios | | --- | --- | --- | --- | --- | | 1.2 Software Design Architecture | 11/6/21 | Done | 11/17/2021 | Josh McGrew | | | | | | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2. Interface Design   | 2.1 User Interface Specifications | 11/6/21 | Done | 12/1/2021 | Mumtahan Firdous | | --- | --- | --- | --- | --- | | | | | | | 3. Database Design   | 3.1 ER Design | 11/6/21 | Done | 12/1/2021 | Trevor Gruber | | --- | --- | --- | --- | --- | | 3.2 Creating the Database | 11/6/21 | Done | 12/1/2021 | Trevor Gruber | | | | | | | 4. Component Design   | 4.1 Component Design Specification | 11/6/21 | **Updated by Trevor and Mumtahan** | 12/1/2021 | Juan Varela Ortiz | | --- | --- | --- | --- | --- | | | | | | | 5. Software Requirements Traceability Matrix | 11/6/21 | Done | 12/1/2021 | Gavin Worley | | | | | |

## Iteration-3

**Overview**

For Iteration-3 we have been working on both front-end and back-end. As we have 6 people in our group, we split among us, 3 in each group. In the front-end group, we have Mumtahan Firdous, Josh McGrew, and Nick Rios. And in the back-end group, we have Gavin Worley, Juan Varela, and Trevor Gruber.

So far we have created a sample college website and back-end functionalities. For the backend, we used Python with Surprise, Pandas, Flask, and MySQL. And for the front-end, we used HTML, CSS, and JavaScript.

For the front-end so far we have the basic website ready. With 14 web pages, including the search option and a separate login webpage. Most of the web pages are completed, yet a couple of them need further development. Besides that, we have a Non-Personalized webpage that has 2 different sections that list the most frequent search terms and keywords.

For the back-end, we’ve made progress on the API functionality. So far, the login functionality is complete. For personalized recommendations, we have taken a few steps in the right direction, but it is not currently functional. We’ve made a few test cases to help define how we want to test our system.

We also wrote our abstract for the scholarship expo and then edited it down to be less than 250 words long.

**Overall Tasks:**

| **Task** | **Assigned Date** | **Completed** | **People** |
| --- | --- | --- | --- |
| Front End | 1/23/2022 |  | Nicolas Rios, Josh McGrew, Mumtahan Firdous |
| Back End | 1/23/2022 |  | Gavin Worley, Juan Varela, Trevor Gruber |
| Testing | 1/23/2022 |  | Work In Progress |

**Back End Breakdown:**

| **Function** | **Assigned Date** | **Completed** | **People** |
| --- | --- | --- | --- |
| Connect to Database | 1/25/2022 |  | Trevor Gruber |
| Login | 1/27/2022 |  | Trevor Gruber |
| Sample Data | 2/13/2022 |  | Trevor Gruber |
| Unit Tests | 2/13/2022 |  | Trevor Gruber |
| Keyword Search | 2/17/2022 |  | Trevor Gruber |
| Personalized Recommendation | 1/27/2022 |  | Gavin Worley |
| Non-Personalized Recommendation | 2/7/2022 |  | Juan Varela |
| Add History Entry | 1/27/2022 |  | Juan Varela, Trevor Gruber |

**Front End Breakdown:**

| **Task** | **Assigned Date** | **Updated** | **Completed** | **People** |
| --- | --- | --- | --- | --- |
| Homepage | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| About | 2/5/2022 |  |  |  |
| Contacts | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Major | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Minor | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Books | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| History | 2/5/2022 |  |  |  |
| Scholarships | 2/5/2022 |  |  | Josh McGrew |
| Admissions | 2/5/2022 |  |  |  |
| Staff | 2/5/2022 |  |  |  |
| Account/Settings | 2/5/2022 | 2/17/22 |  | Nick Rios |
| Login/Logout | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Non-personalized | 2/13/2022 |  |  | Mumtahan Firdous |

**Personal thoughts so far:**

* Gavin Worley: I have made some positive progress since the last meeting on the personalized recommendations. Starting with the ability to connect to and pull data from the local database, and working on fitting a system to our test data. I am currently using pandas and surprises for all of my work. Also, since I was initially having trouble connecting to the database I decided to try working with the .csv files I exported from MySQL to test my code.
* Juan Varela: Overall the prototype seems to be making good progress. The front-end and back-end have both reached significant milestones save for being connected to each other.
* Trevor Gruber: I think we’re making good progress, we just need to test, polish the system, and finally connect the backend and frontend. Connecting the front and back end will likely require either JavaScript or PHP.
* Nicolas Rios: I believe that the development of the prototype is going well, as we have made significant advancements in both the front-end and back-end. Although we still need to implement code to communicate requests from the front-end to the database in the back-end, the overall functionality is coming together.
* Josh McGrew:
* Mumtahan Firdous: So far I think we doing good. Made some progress, but still, need to work on some stuff. But hopefully, we’ll figure by the end of the upcoming week.

## Iteration-4

**Overview**

For Iteration-4 we have still been working on both front-end and back-end. As we have 6 people in our group, we split among us, 3 in each group. In the front-end group, we have Mumtahan Firdous, Josh McGrew, and Nick Rios. And in the back-end group, we have Gavin Worley, Juan Varela, and Trevor Gruber.

Besides the frontend and backend, we have also been working on the STD - Software Testing Document. We divided the parts among ourselves and helped each other to finish it.

We have also been working on the Poster and finalizing the abstract for the Expo. We are also registered for the Scholarship Expo and will be printing out the poster next week.

**Back End Breakdown:**

| **Function** | **Assigned Date** | **Completed** | **People** |
| --- | --- | --- | --- |
| Connect to Database | 1/25/2022 |  | Trevor Gruber |
| Sterilize Inputs | 1/25/2022 |  | Trevor Gruber |
| Login | 1/27/2022 |  | Trevor Gruber |
| Recommendation | 1/27/2022 |  | Gavin Worley |
| Add History Entry | 1/27/2022 |  | Juan Varela, Trevor Gruber |

**Front End Breakdown:**

| **Task** | **Assigned Date** | **Updated** | **Completed** | **People** |
| --- | --- | --- | --- | --- |
| Homepage | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| About | 2/5/2022 | 2/18/22 |  | Nick Rios |
| Contacts | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Major | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Minor | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Books | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| History | 2/5/2022 | 3/3/22 |  | Nick Rios |
| Scholarships | 2/5/2022 | 2/10/22 |  | Josh McGrew |
| Admissions | 2/5/2022 | 2/24/22 |  | Josh McGrew |
| Staff | 2/5/2022 | 2/18/22 |  | Mumtahan Firdous |
| Account/Settings | 2/5/2022 | 2/17/22 |  | Nick Rios |
| Login/Logout | 2/5/2022 | 2/8/22 |  | Mumtahan Firdous |
| Search   | Personalized | 2/19/2022 | 2/29/22 |  | Mumtahan Firdous | | --- | --- | --- | --- | --- | | Non-Personalized | 2/19/2022 | 2/29/22 |  | | | | | |

## Iteration-5

**Overview**

We started Iteration 5 by participating in the Bradley Scholarship Expo by presenting our poster and a demo.

The EXPO, overall, went quite well. We were given 10 minutes to present in front of two judges. We divided the presentation into 6 points: Motivation and Problems, Background, Our Goals, Our Approaches, Conclusion, and Future Work. Besides these 6 points, we also had use cases for the users. One of the judges thought signing into the websites can be a little hassle. And forcing the users to sign in to the website just to search can cause user loss.

We have been finishing implementation and testing. We needed to do a lot of Ajax connections to make the website usable. We also created the user manual. It consists of documentation of the software designed to help users.

The last thing we have left is working on the final report, which will mostly consist of appendices that contain our earlier reports, such as SPMP, SDD, SRS, STD, and so on. This final report will go over each iteration, discussing what our group accomplished in each.

**Meetings:**

* 3/27: We discussed the test cases, and what we need to update and change.
* 3/31: We discussed the EXPO, our poster is printed, and we are working on the presentation.
* 4/3: Didn’t have much to discuss, went over the EXPO and talked about the demo.
* 4/12: Worked on the test cases, and updated the report.
* 4/14: Went over what we did if there is anything else that needs an update, meeting tomorrow in person at 10 am.
* 4/19: We discussed how to create the user manual, and worked on it.
* 4/24: We discussed how we should update the search option, we need to fix the registration page and work on some test cases, they need to be slightly modified. The search option needs to give URL outputs not only keywords.
* 4/28: This is our last meeting before the final presentation. Found any errors we had and updated the final presentation, user manual, and the reports.



# 

# 

# 

# 

# 

# 

# 

# **Developer’s Profile**