<MentionNotifier>

Requirements Specification and Analysis

<1.0>

<06/01/2018>

<Berk Ergun>

Prepared for

CSE490 Thesis Project



Table of Contents

[1. Introduction 1](#_Toc503095025)

[1.1. Purpose of the System 1](#_Toc503095026)

[1.2. Scope of the System 1](#_Toc503095027)

[1.3. Objectives and Success Criteria of the Project 1](#_Toc503095028)

[1.4. Definitions, Acronyms, and Abbreviations 1](#_Toc503095029)

[1.5. Overview 1](#_Toc503095030)

[2. Current System 1](#_Toc503095031)

[3. Proposed System 2](#_Toc503095032)

[3.1. Overview 2](#_Toc503095033)

[3.2. Functional Requirements 2](#_Toc503095034)

[3.3. Nonfunctional Requirements 2](#_Toc503095035)

[Usability 2](#_Toc503095036)

[Reliability 3](#_Toc503095037)

[Performance 3](#_Toc503095038)

[Supportability 3](#_Toc503095039)

[Implementation 3](#_Toc503095040)

[Interface 3](#_Toc503095041)

[Packaging 3](#_Toc503095042)

[Legal 3](#_Toc503095043)

[3.4. System Models 3](#_Toc503095044)

[Scenarios 4](#_Toc503095045)

[Use case model 6](#_Toc503095046)

[Object model 9](#_Toc503095047)

[Dynamic model 9](#_Toc503095048)

[User interface—navigational paths and screen mock-ups 10](#_Toc503095049)

[3.5. Project Schedule 12](#_Toc503095050)

[4. Glossary 13](#_Toc503095051)

[5. References 13](#_Toc503095052)

REQUIREMENTS ANALYSIS DOCUMENT[1]

# Introduction

## Purpose of the System

The purpose of the system is to allow for its user to keep up with things of his/her interest over the internet through a desktop application.

## Scope of the System

The project’s software will be a desktop application and will be used by a single user. The applications will try not to reinvent the wheel and use an existing search engine for its purposes. The project includes three reports and a final thesis report about the project and its development. The three reports include this document, the Requirements Analysis Document, System Design Document and an Object Design Document. The Thesis Document can be thought of as a collection of these three documents. The project will be delivered by the end of this semester.

## Objectives and Success Criteria of the Project

The project will be deemed successful when it's delivered by the end of this semester and is able to perform its Functional Requirements while abiding by its Non-functional Requirements.

## Definitions, Acronyms, and Abbreviations

UI: User Interface

## Overview

This document first contains a comparison of currently existing alternative systems and the proposed system. Then goes on to the system’s functional and non-functional requirements. In section 3.4, we take a look at the system models, use cases and scenarios involving our system. This section also includes the proposed UI mockups and a Gannt chart for the project’s schedule. The document ends with the glossary and references.

# Current System

This is a greenfield project and is unrestricted by an existing system and its constraints. In comparison to other system available, our system looks to be an automated, concurrent system, allowing for its user to make searches of his/her interest, in multiples at a same time and see updates compared to its previous iterations

# Proposed System

A desktop application where the user can make concurrent, repeating searches on keywords he/she provides and view their results within the it.

## Overview

The system will be a desktop application where the user can provide search keywords for the application and choose which to include before searches. The user will be able to specify its search intervals and the 'depth' of each search, which is the number of page results from the web search engine used. The user then will be able to list the results of these searches similar to how the web search engine lists it's results on a web browser. Each result will have an indicator, showing whether they were updated since the last iteration of a search.

## Functional Requirements

* The system should allow for its user to do repeated web searches.
* The system should allow for its user to provide search keywords.
* The system should provide a way for its user to remove search keywords if they wish to.
* The system should allow for its user to provide multiple search keywords.
* The system should allow for its user to combine these keywords in any way they wish.
* The system should allow for its user to indicate search intervals for a search.
* The system should allow for its user to indicate the depth of a search.
* The system should allow for its user to view the results of these searches.
* The system should allow for its user to view the website for a search result on a web browser.
* The system should provide a way for its user to access different, currently-running searches' results.
* The system should allow for its user to terminate a currently running search task.
* The system should allow for its user to view updated search results without having for him/her to go through all the results.
* The system should notify its user when a search is complete or new results are available.

## Nonfunctional Requirements

### Usability

* The system should be simple and easy to use for about any user that has used a web search engine previously.
* A user should be able to access the website of a search results through the system.
* The system results should be viewed in a familiar fashion to the user.
* The system should be barebones, in the sense that its user should be able to use its main purpose, make searches, without having to register to the system or anything.
* The system should impose no constraint on its user through an external administration or management or anything.

### Reliability

* The system should reliably make searches without interfering with other ongoing search tasks.
* The system should be able to perform its purposes as long as it’s used search engine, Google, provides the service to it.

### Performance

* The system should be able to handle many ongoing search tasks and should only be limited by its user's running computer setup.
* While the searches are assumed to be background tasks for the user, they should be still fast in the way they perform.
* The system should not be bloated as it runs and not interfere with its user's other tasks.

### Supportability

* As a discrete, standalone system with no further support provided, it should be robust as possible.
* As no further support will be provided, its code should be able to be modified by its user if they deem themselves to be eligible to do so.

### Implementation

* The system should a desktop PC application.
* The system should use the Google Search Engine.
* The system should be created with Windows Operating Systems in mind.
* The system should be written in the Python Programming Language.

### Interface

* The system should be able to handle Google Search Engine results and queries.

### Packaging

* The system should be set as Python module that can be run through the command line (even though it doesn't necessarily need it).

### Legal

* As the system is using the Google Search Engine, it is bound to whatever constraints it might impose in viewing search results.

## System Models

Describes the scenarios, use cases, object model, and dynamic models for the system. This section contains the complete functional specification, including mock-ups illustrating the user interface of the system and navigational paths representing the sequence of screens.

### Scenarios

|  |  |
| --- | --- |
| *Scenario name* | **searchingAboutRobocop** |
| *Participating actors instances* | bob:User |
| *Flow of events* | 1. Bob is a Robocop fan and has heard that a new Robocop film is in the making. Bob wants to know when there is new information is available about it.  2. Bob fires up MentionNotifier and enters his keyword, 'Robocop' and presses the 'Add Keyword' button.  3. Bob thinks this search might be too broad so he also enters 'new movie' and again, presses 'Add Keyword'.  4. Thinking this is enough, Bob picks the two keywords from the list and hits the 'Start Search' button and waits for the results to come in.  5. With the search's complete, a notification appears, informing Bob that the search is now complete. Bob enters the 'Results' tab and clicks the 'Robocop new movie' button to view its results.  6. A list of results appear, showing the results of the search but they show no indication of being updated as this is the first iteration of this search.  7. After a while, another notification appears, informing Bob that there are 7 updates on his search. Bob again, clicks the 'Robocop new movie' button to refresh the viewed results and as promised, sees that a couple of websites have been updated since the last search.  8. Bob presses the 'Float Updated' button to bring up the updated results  9. Bob presses an updated search result's link address to open it in his web browser and reads the new movie. A new notification informing him of 3 new results appear, Bob is satisfied. |

|  |  |
| --- | --- |
| *Scenario name* | **searchingAboutMargeretAndTracy** |
| *Participating actors instances* | lucy:User |
| *Flow of events* | 1. Lucy is an avid reader of fantasy literature genre and learns that Margeret Weis and Tracy Hickman, longtime writer duo, will be releasing two new books, written separately by each.  2. Lucy starts the MentionNotifier application and enters the keywords, 'Tracy Hickman', Margeret Weis' and 'new book'.  3. Lucy picks 'Tracy Hickman' and 'new book' from the list by clicking on them and presses the 'Start Search' button.  4. While that search is ongoing, Lucy disables the 'Tracy Hickman' keyword by again clicking on it and this time, clicks on 'Margeret Weis'.  5. Lucy again, clicks on the 'Start Search' button and goes on to the Results Tab to wait for her results and sure enough, a notification pops up informing her that her search on 'Tracy Hickman new book' is done.  6. Lucy clicks the 'Tracy Hickman new book' button under the 'Ongoing Searches' list and is given the results of her search.  7. While Lucy is going through the results, a new notification appears, again, informing her that her search on 'Margeret Weis new book' is now complete.  8. After some time while Lucy is going through her first search results, another notification appears, informing her that there are 2 new updates on the 'Tracy Hickman new book' search. Lucy again, clicks the 'Tracy Hickman new book' button and refreshes the results. Two results previously not indicated is now marked as 'Updated'. Lucy clicks one their links to reach the website through her web browser.  9. With the newly updated site open in her web browser, Lucy goes through the information while a notification pops up informing her of a newly updated website about her previous 'Margeret Weis new book' search. Lucy is satisfied. |

|  |  |
| --- | --- |
| *Scenario name* | **modifiedSearch** |
| *Participating actors instances* | mark:User |
| *Flow of events* | 1. Mark is a working man with assets in Bitcoin. He is not frantic about it but still wants to learn about how Bitcoin is performing from time to time.  2. Mark starts the MentionNotifier application and enters the keyword 'Bitcoin' and presses Enter.  3. Mark is not really interested about Etherium, another popular cryptocurrency so he types in '-Etherium' into the text entry field and presses enter. He knows that this will disable results with Etherium in it.  4. Mark is currently working and does not want to be flooded with notifications from time to time and sets the search intervals at 2 hours. He is also interested in the most prominent of news so he sets the 'Number of Pages' at 3.  5. Mark presses the 'Start Search' button and shortly after, is informed that the search is now complete. He doesn't check the results and continues on with his work.  6. After two hours, as set, a notification appears, informing Mark that there are 3 websites updated. Mark is satisfied. |

|  |  |
| --- | --- |
| *Scenario name* | **reconsideringSearches** |
| *Participating actors instances* | selin:User |
| *Flow of events* | 1. Selin is an avid Justin Bieber fan. So much so that she wants to know anything new about him. She is following his information through all social media networks and wants to also do this for information appearing in Google search results.  2. Selin starts the Mention Notifier application and enters 'Justin Bieber' as her keyword.  3. She sets the search interval at 10 minutes and sets the Number of Pages at 0. She knows that this will force the application to retrieve all pages results available about Justin Bieber.  5. She starts the search by clicking the Start Search button but then thinks that she doesn't really need information about Selena Gomez with her Justin. She goes on the Results Tab and picks the 'Justin Bieber' search ongoing and presses 'Stop Search' button.  6. She goes back on the Search Tab and enters the keywords '-selena' and '-gomez'. After picking the two newly added keywords, she starts the search again by pressing the 'Start Search' button.  7. After a sometime, a notification appears informing her that the search is now complete. Selin goes on the 'Results Tab' and presses the 'Justin Bieber -selena -gomez' result listed.  8. She presses the 'List Results' button and is given the hundreds of results of the search she just did, all with information about Justin and no Selena. Selin is satisfied. |

### Use case model

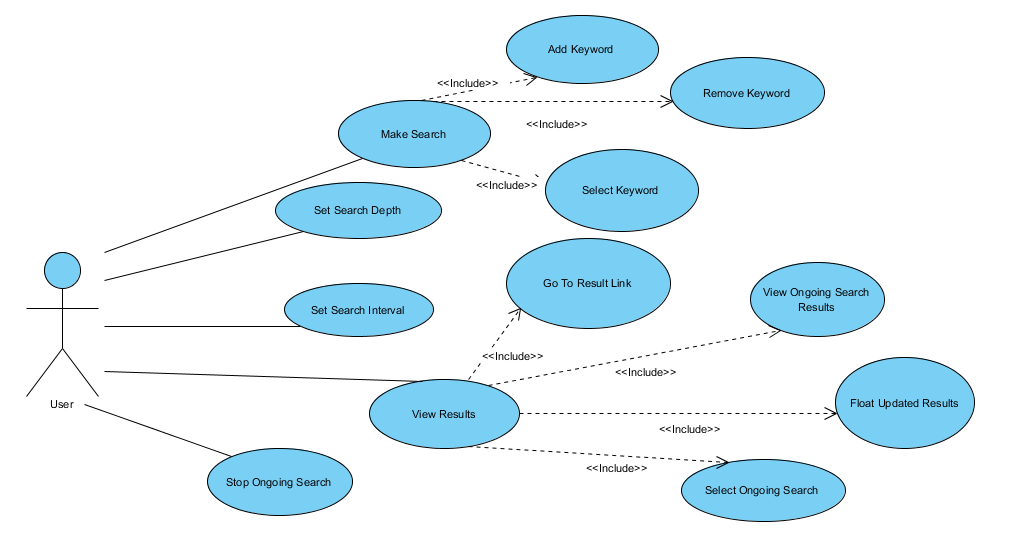


Diagram 1 - System Use Case Diagram

|  |  |
| --- | --- |
| *Use case name* | **MakeSearch** |
| *Participating actors* | Initiated by User |
| *Flow of events* | 1. User enters his/her keyword into the keyword entry field. 2. User presses the ‘Add Keyword’ button. 3. MentionNotifier adds the keyword into the Keywords list.    1. User can add more keywords if he/she wishes to do so. 4. User clicks on the keyword(s) he/she wishes to include in his/her search. 5. MentionNotifier marks the keyword as selected and changes their color to indicate that they are selected. 6. User presses the ‘Start Search’ button.    1. User can modify the Search properties by changing the ‘Number of Pages’ and ‘Search Interval’ values. 7. MentionNotifier starts a search with given keywords and search properties. |
| *Entry condition* | * User is in the ‘Search Tab’. * User has already added a keyword. (**AddKeyword**) |
| *Exit condition* | * User successfully starts a search and confirms it in the ‘Results Tab’ under the ‘Ongoing Searches’ tab. |
| *Quality requirements* | * Any number of keywords can be added. * User can set the ‘Number of Pages’ to 0 to retrieve all pages of a search’s results. * Any number of searches can be done concurrently. |

|  |  |
| --- | --- |
| *Use case name* | **AddKeyword** |
| *Participating actors* | Initiated by User |
| *Flow of events* | User enters his/her keyword into the keyword entry field.  User presses the ‘Add Keyword’ button.  MentionNotifier adds the keyword into the Keywords list.  User can add more keywords if he/she wishes to do so. |
| *Entry condition* | * User is in the ‘Search Tab’. |
| *Exit condition* |  |
| *Quality requirements* | * If there is nothing entered in to the keyword entry field, pressing enter or the ‘Add Keyword’ does nothing. |

|  |  |
| --- | --- |
| *Use case name* | **RemoveKeyword** |
| *Participating actors* | Inititated by User |
| *Flow of events* | 1. User clicks on the keyword(s) he/she wants to remove. 2. MentionNotifier marks the keyword as selected and changes their color to indicate that they are selected. 3. User clicks the ‘Remove Keyword’ button. 4. MentionNotifier removes the selected keywords form the list of ‘Keywords’. |
| *Entry condition* | * User has already added a keyword. (**AddKeyword**) |
| *Exit condition* | * User confirms that the keyword he/she wanted to remove is not under the ‘Keywords’ list anymore. |
| *Quality requirements* | * If there are no keywords selected, the button does nothing. |

|  |  |
| --- | --- |
| *Use case name* | **ViewResults** |
| *Participating actors* | Initiated by User |
| *Flow of events* | 1. User clicks the ‘Results Tab’ tab at top. 2. MentionNotifier changes the viewed tab to the ‘Results Tab’. 3. User clicks his/her search under the ‘Ongoing Searches’ list, listed by its combined keywords. 4. MentionNotifier marks the search as selected and changes its color to indicate that its selected. 5. User presses the ‘View Results’ button. 6. If the search is done, MentionNotifier shows the search results for the selected search. Otherwise, it does nothing. |
| *Entry condition* | * User is in the ‘Search Tab’. * User has performed a search. (**MakeSearch**) |
| *Exit condition* | * User confirms that the search’s results are being shown. |
| *Quality requirements* |  |

|  |  |
| --- | --- |
| *Use case name* | **StopOngoingSearch** |
| *Participating actors* | Initiated by User |
| *Flow of events* | User clicks the ‘Stop Search’ button.  MentionNotifier stops the currently selected search.  MentionNotifier removes the search from the list of ‘Ongoing Searches’. |
| *Entry condition* | * User has viewed a search’s results. (**ViewResults**) |
| *Exit condition* | * User confirms that the search he/she wanted to stop is no longer under the list of ‘Ongoing Searches’. |
| *Quality requirements* |  |

|  |  |
| --- | --- |
| *Use case name* | **GoToResultLink** |
| *Participating actors* | Initiated by User |
| *Flow of events* | 1. User clicks on the link for a search result. 2. MentionNotifier opens the user’s default web browser application to view the webpage. |
| *Entry condition* | * User has viewed a search’s results. (**ViewResults**) |
| *Exit condition* | * User reaches his/her web browser with the link. * User has no default web browser. |
| *Quality requirements* |  |

|  |  |
| --- | --- |
| *Use case name* | **FloatUpdatedResults** |
| *Participating actors* | Inititated by User |
| *Flow of events* | 1. User presses the ‘List Results’ button to refresh the results if no updated result is being showed in results. 2. MentionNotifier views the search results. If results are different from what is shown (if an earlier iteration of search’s results are being shown), they are discarded. 3. User presses the ‘Float Updated’ button. 4. If there are results marked as updated, MentionNotifier lists the at the top of the results listed. |
| *Entry condition* | * User has viewed a search’s results. (**ViewResults**) * MentionNotifier has performed more than one iteration of searches. |
| *Exit condition* |  |
| *Quality requirements* |  |

### Object model

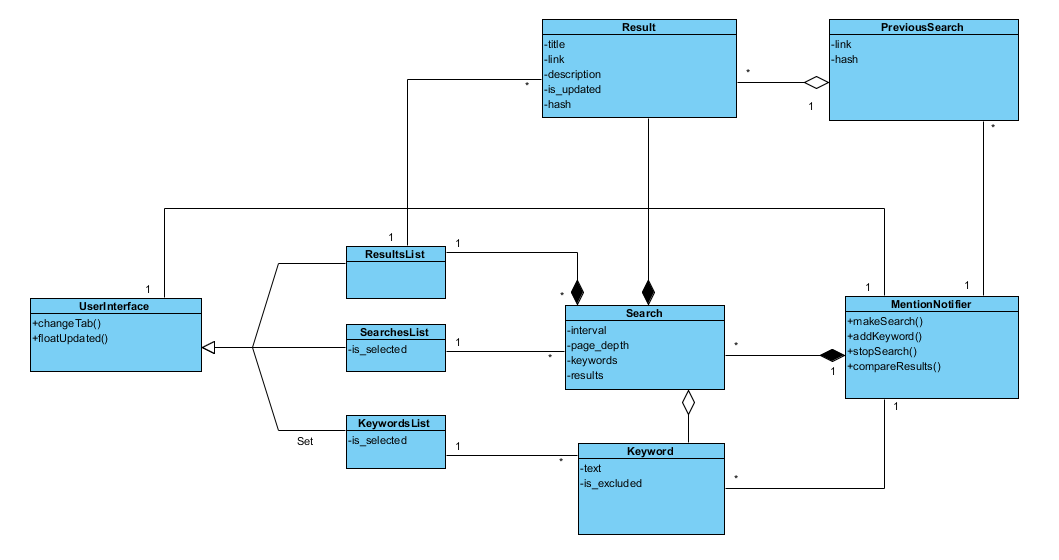


Diagram 2 - Object Diagram

### Dynamic model

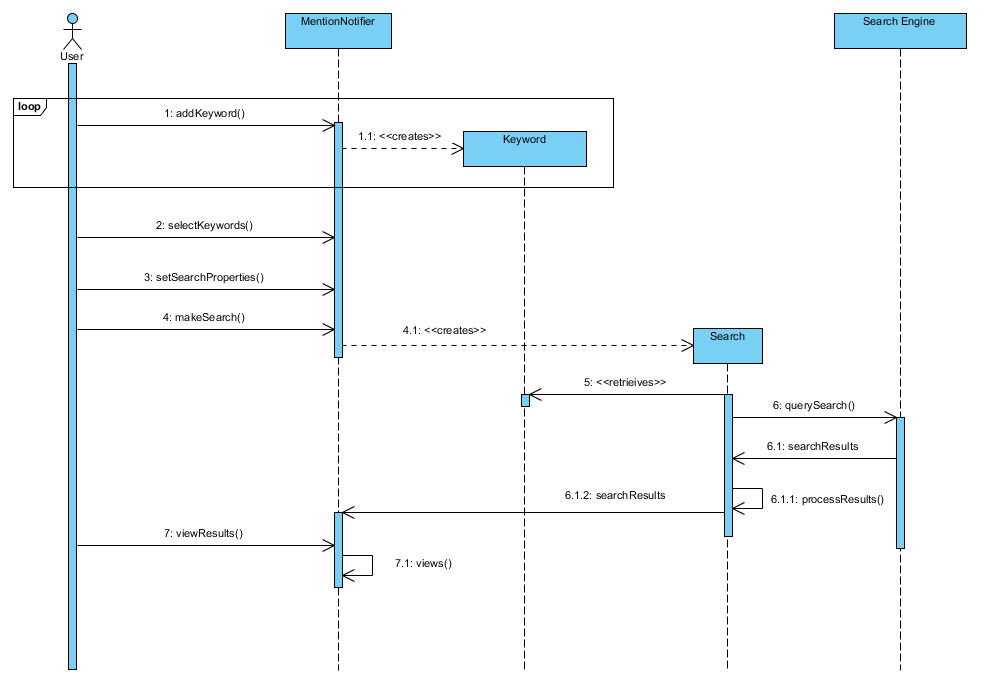
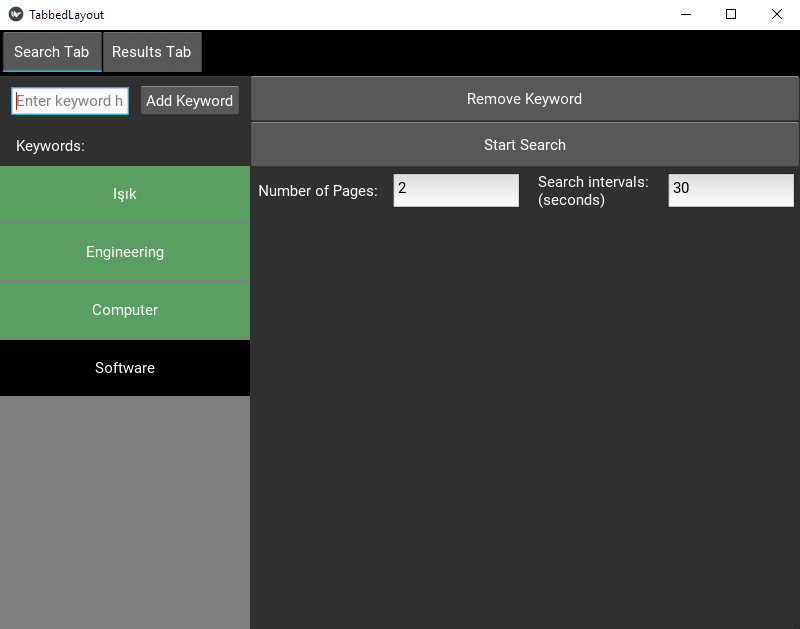


Diagram 3 - Sequence Diagram for Common Use

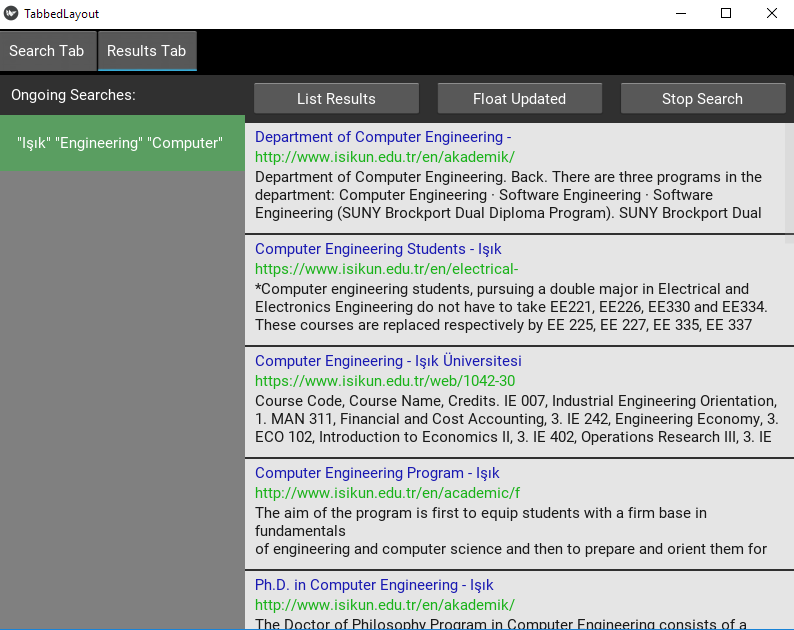
The dynamic model is depicted with sequence diagrams and with state machines. Sequence diagrams represent the interactions among a set of objects during a single use case. State machines represent the behavior of a single object (or a group of very tightly coupled objects). The dynamic model serves to assign responsibilities to individual classes and, in the process, to identify new classes, associations, and attributes to be added to the analysis object model.

When working with either the analysis object model or the dynamic model, it is essential to remember that these models **represent user-level concepts, not actual software classes or components.**

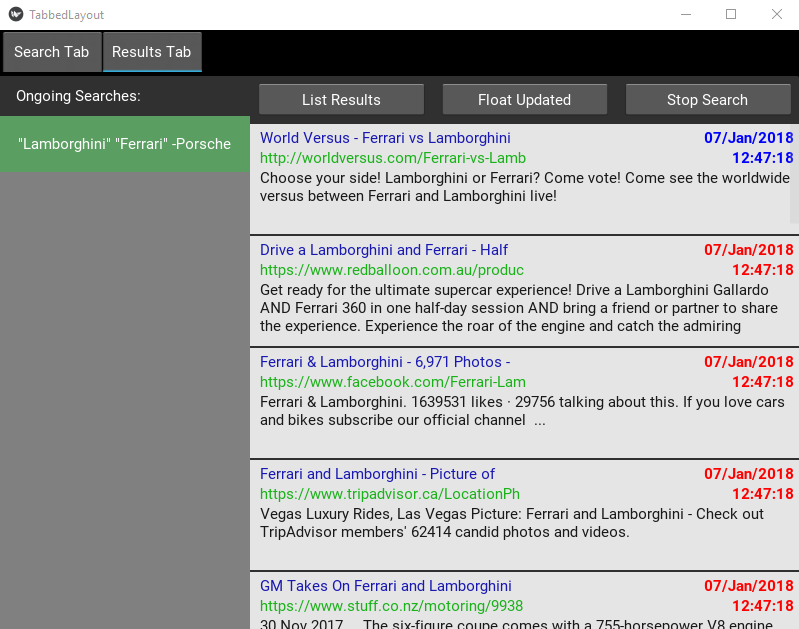
### User interface—navigational paths and screen mock-ups



Mockup 1 – Search Tab



Mockup 2 – Results Tab (No Update)



Mockup 3 – Results Tab (With Updates – blue indicates already viewed results)

## Project Schedule

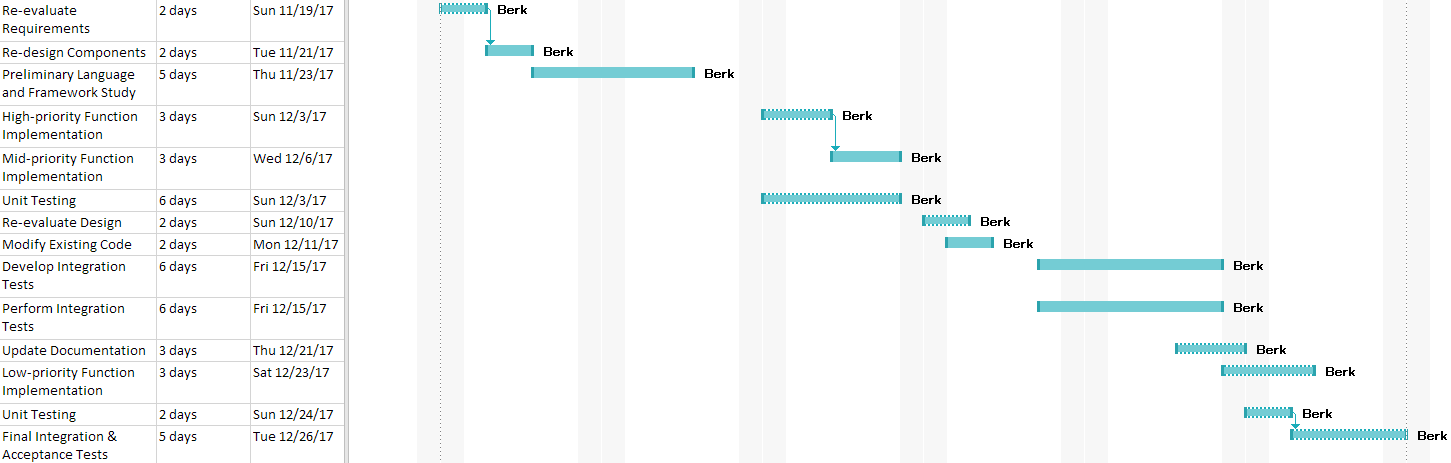


Table 1 – Project Schedule (Gantt Chart)

# Glossary

User: Any person who may be using the system.

Keyword: A word or group of words provided by the User for a search task.

Search: A search task initiated by a User, provided with one or many keywords and set search properties.

Hash: Data returned by a cryptographic hash function.

Results: Data returned by the Web Search Engine, related to the provided keywords.

Search Interval: The time between each iteration of a search.

Page Depth: Pages of results to be retrieved from the Web Search Engine.

Web Search Engine: Software system that is designed to search for information on the World Wide Web. [3]

# References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. “Nonfunctional Requirement Examples.” *Requirements Quest*, requirementsquest.com/nonfunctional-requirement-examples/.
3. “Web search engine.” *Wikipedia*, Wikimedia Foundation, 4 Jan. 2018, en.wikipedia.org/wiki/Web\_search\_engine.