

Florida International University
School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Automated Document Summarization 1.0

Team Members: Alastair Paragas and Alberto Mizrahi

Product Owner(s): Nagarajan Prabakar

Mentor(s): Nagarajan Prabakar

Instructor: Masoud Sadjadi

The MIT License (MIT)
Copyright (c) 2017 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction,

including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

Automated Document Summarization 1.0, more succinctly known as Tidbit, is a web application that utilizes advanced natural language processing (NLP) algorithms to extract the textual information of an online document and then summarize it by selecting the sentences that capture the main points of its content. This application was developed in response to the need of individuals to constantly read various articles to keep their knowledge up to date. This, however, requires a considerable investment of time and energy. By being able to concisely synopsise online articles, Tidbit allows these individuals to achieve their goal of constantly being informed in a swift and efficient manner.

The following documentation will explain the design, development and implementation choices that were made during the course of the development of the application. This includes descriptions of the developed system, related software development framework documentation, system design decisions, system validation suites as well as a variety of other related information regarding Tidbit's functionality and design.

Table of Contents

INTRODUCTION

CURRENT SYSTEM	6
PURPOSE OF NEW SYSTEM	6

USER STORIES

IMPLEMENTED USER STORIES	7
PENDING USER STORIES	11

PROJECT PLAN

HARDWARE AND SOFTWARE RESOURCES	15
SPRINTS PLAN	17
<i>Sprint 1</i>	17
<i>Sprint 2</i>	17
<i>Sprint 3</i>	17
<i>Sprint 4</i>	18
<i>Sprint 5</i>	19

SYSTEM DESIGN

ARCHITECTURAL PATTERNS	19
SYSTEM AND SUBSYSTEM DECOMPOSITION	21
DEPLOYMENT DIAGRAM	21
DESIGN PATTERNS	21

SYSTEM VALIDATION

GLOSSARY

APPENDIX

APPENDIX A - UML DIAGRAMS	31
<i>Static UML Diagrams</i>	31
<i>Dynamic UML Diagrams</i>	40
APPENDIX B - USER INTERFACE DESIGN	44
APPENDIX C - SPRINT REVIEW REPORTS	49

APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST
DOCUMENT AND OTHER DOCUMENTS..... 51

REFERENCES69

INTRODUCTION

In this section, a brief overview of the Tidbit application will be presented. First, a description of the current system will be given. Thereafter, an explanation of the new system will be presented, including the intended purpose for which it was developed.

Current System

This was the first iteration in the development of Tidbit. In other words, there was no system currently in place and the entire application was built from the ground up.

Purpose of New System

During the course of a day, individuals are constantly bombarded with information from various different angles: news sites, emails, social media, and many others. Furthermore, to stay up to date with a subject of interest, he or she must invest an exorbitant of time, energy and effort to read all this information. Considering a person's daily responsibilities, this is at the very least exhaustive and at worst, it is simply impossible.

Tidbit was developed to alleviate this issue. Given a URL that points to a particular news article or online document, this web application will utilize a variety of Natural Language Processing (NLP) algorithms to achieve the following:

1. **Extract the content of the article in the website:** This is not a trivial task since most websites have a barrage of advertisements, links and texts that have no connection to the content of the article itself.
2. **Tokenize the text of the article:** Given the complexities and nuances of the English language, developing an algorithm that break up a text into its constituent sentences is not simple.
3. **Summarize the document:** In this step, the most significant and descriptive sentences in the text are selected. These sentences must capture the essence of the document.

For the last step, Tidbit supports three of the most well-known NLP summarization algorithms: straightforward frequency-based, Luhn's algorithm (Luhn, 1958) and the SumBasic (Vanderwende *et al.*, 2007) algorithm.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the Automated Document Summarization project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

The following user stories were implemented during the course of this iteration:

1. Understand the use of Streams with Haskell and Conduit
2. Develop a Linear Regression application
3. Set up the basic back-end API
4. Understand SVMs and develop applications for cross-validation
5. Create authentication routes
6. Create summarization routes
7. Implement summarization algorithms
8. Design the front end and deploy the website
9. Implement another summarization algorithm

We now present the details of all the implemented user stories.

User Story #1: Understand the use of Streams with Haskell and Conduit

Description:

- As a Programmer, I need to be confident with using Haskell and streaming libraries like Conduit so that we understand how they work given that we will be using Haskell as one of the two main programming languages in this project and we are potentially going to be creating streams of data flow into the system.
- As a Researcher, I would like to be able to see streaming Twitter tweets on a console because we want to verify that a system could be built that allows for streaming data.

Acceptance Criteria:

- Have a sample Twitter feed implementation using the Conduit Library and Haskell.

User Story #2: Develop a Linear Regression application**Description:**

- As a Researcher, I would like for the team to go through Andrew Ng's Lecture Notes 1 and 2 from his ML course in Stanford so that they should obtain a deeper understanding of the theory behind the technologies they will be using for this project.
- As a Researcher, I would like to see Linear Regression implemented so that we can predict continuous vectors of numbers.

Acceptance Criteria:

- The team should have a copy of the research notes they took from the lectures.
 - The team should have implemented the linear regression algorithm.
 - They team must have developed an application that can successfully execute linear regression on a set of data.
-

User Story #3: Set up the basic back-end API**Description:**

- As a Researcher, I would like for the team to develop the basic structure of the back-end API (which includes support for user registration, login, logout, etc.) so that the application's foundation will be working correctly.

Acceptance Criteria:

- The team must have chosen and set up the libraries and frameworks that they will use to develop the back-end.
- The team must have added support in the API for the User resource, with the ability to create, delete, get and update users in the system.
- The team must have created documentation explaining the functionality added to the API.

User Story #4: Understand SVMs and develop applications for cross-validation**Description:**

- As a Researcher, I would like for the team members to go through Andrew Ng's Lecture Notes 3, 4 and 5 from his ML course in Stanford so that they should obtain a deeper understanding of the theory behind the technologies they will be using for this project.
- As a Researcher, I would like for the team to develop an application that conducts cross-validation for model selection of learning algorithms so that they will have this important tool available when they begin developing their Machine Learning algorithms.

Acceptance Criteria:

- The team should have a copy of the research notes they took from the lectures.
 - The team should have developed an application that implements the main cross-validation algorithms: Hold-out CV, k-fold CV and Leave-one-out CV.
-

User Story #5: Create authentication routes**Description:**

- As a User, I would like to be able to login into the application so that I can make sure that my session is private.

Acceptance Criteria:

- The team should have implemented the /authenticate api route.
 - The team should have unit tested the /authenticate api route.
-

User Story #6: Create summarization routes**Description:**

- As a User, I would like for the application to keep track of the documents I have summarized in the past so that I can check this history any time I wish.
- As a Data Analyst, I would like for the team to keep track of the documents that the system has summarized so far so that we can conduct statistical tests on this data to obtain insights on our users activities and preferences.

Acceptance Criteria:

- Implemented summarization API routes for documents and history resources.
 - Unit tested summarization API routes for documents and history resources.
-

User Story #7: Implement summarization algorithms**Description:**

- As a User, I would like to be able to summarize an article so that it saves me the time to have to read through its entirety.

Acceptance Criteria:

- The team must have a working version of a summarization algorithm.
 - The team must have tested this algorithm as thoroughly as possible.
-

User Story #8: Design the front end and deploy the website**Description:**

- As a User, I would like that the UI of the application should be designed as a professional web app so that it is easy as well as enjoyable to use.
- As a User, I would like for the application to be available online so that I can access it wherever I go.

Acceptance Criteria:

- The team must have developed a front end UI for the application.
 - The front end must be hooked up to the back end API, and hence the entire application should be working correctly.
 - The website must be deployed to a live production server.
-

User Story #9: Implement another summarization algorithm**Description:**

- As a User, I would like that the application allow me to choose from various summarization algorithms when summarizing a document so I can see which one works best for me.

Acceptance Criteria:

- The team should implement another summarization algorithm besides the frequency one they have already implemented.
- The team must have tested this algorithm as thoroughly as possible.

Pending User Stories

The following are the user stories whose implementations are still pending and which should be considered for the next iteration of this project:

1. Implement the k-means clustering algorithm
2. Implement the Principal Component Analysis (PCA) algorithm
3. Implement the Value Iteration algorithm
4. Encrypt stored passwords
5. Improve the text extraction algorithm
6. Implement advanced summarization algorithms

We now present the details of all the pending user stories.

User Story #1: Implement the k-means clustering algorithm

Description:

- As a Researcher, I would like for the team to go through Andrew Ng's Lecture Notes 6, 7a, 7b and 8 from his ML course in Stanford so that they should obtain a deeper understanding of the theory behind the technologies they will be using for this project.
- As a Researcher, I would like to see the k-means clustering algorithm implemented so that we can group the data into few cohesive clusters, which will provide more insights into how the data is structured and patterned.

Acceptance Criteria:

- Every team member should have read lectures notes 6, 7a, 7b and 8.
 - Every team should have taken notes from his readings to ensure he or she has understood the material.
 - The team must have finished an application that implements the k-means clustering algorithm.
-

User Story #2: Implement the Principal Component Analysis (PCA) algorithm

Description:

- As a Researcher, I would like for the team to go through Andrew Ng's Lecture Notes 9 and 10 from his ML course in Stanford so that they should obtain a deeper understanding of the theory behind the technologies they will be using for this project.
- As a Researcher, I would like to see the Principal Component Analysis (PCA) algorithm implemented so that we can transform a set of observed variables into a smaller set of artificial ones (that still preserve the same information) which would then allow us to process more data in a more efficient manner.

Acceptance Criteria:

- Every team member should have read lectures notes 9 and 10.

- Every team should have taken notes from his readings to ensure he or she has understood the material.
 - The team should have finished an application that implements the Principal Component Analysis (PCA) algorithm.
-

User Story #3: Implement the Value Iteration algorithm

Description:

- As a Researcher, I would like for the team to go through Andrew Ng's Lecture Notes 11 and 12 from his ML course in Stanford so that they should obtain a deeper understanding of the theory behind the technologies they will be using for this project.
- As a Researcher, I would like to see the Value Iteration algorithm implemented so that we can solve finite-state Markov decision processes (MDP), which would allow us to conduct reinforcement learning.

Acceptance Criteria:

- Every team member should have read lectures notes 11 and 12.
 - Every team should have taken notes from his readings to ensure he or she has understood the material.
 - The team should have finished an application that implements the Value Iteration algorithm.
-

User Story #4: Encrypt stored passwords

Description:

- As a User, I would like that the application should store my password in an encrypted manner so that I can be sure that my credentials will remain private in the case of a system's breach.

Acceptance Criteria:

- The passwords in the database should be stored in an encrypted manner.
-

User Story #5: Improve the text extraction algorithm**Description:**

- As a User, I would like for the team to improve the algorithm through which they extract the article's content from its website so that only relevant information is extracted and therefore, the accuracy of the summarization will dramatically improve.

Acceptance Criteria:

- The team must show that the improved algorithm is able to extract an article's main content from various popular news sites.
-

User Story #6: Implement advanced summarization algorithms**Description:**

- As a User, I would like that the application should support more advanced, state-of-the-art Natural Language Processing (NLP) summarization algorithms so that the summaries it produces are more accurate and concise.

Acceptance Criteria:

- The team must have implemented more advanced algorithms such as LexRank (Erkan & Radev, 2004), TextRank (Mihalcea & Tarau, 2004), Latent Semantic Analysis (Steinberger & Jezek, 2009) and KL-Sum (Haghighi & Vanderwende, 2009).

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

Hardware:

- Any computer running Linux, Mac OS or Windows

Software:

- Python v3.6
- Flask v0.12
- Docker v17.06.0-ce
- PostgreSQL v9.6.3
- Pip v3
- Python libraries:
 - Aiohttp v2.1.0
 - Aiopg v0.13.0
 - Aitertools v0.1.0
 - Aniso8601 v1.2.0
 - Async-timeout v1.2.1
 - Asynctest v0.10.0
 - Cerberus v1.1
 - Chardet v3.0.3
 - Click v6.7
 - Flask-RESTful v0.3.5
 - Itsdangerous v0.24
 - Jinja2 v2.9.5
 - MarkupSafe v1.0
 - Multidict v2.1.6
 - Neo4j-driver v1.1.0

- Neomodel v3.2.2
- Psycopg2 v2.7.1
- PyJWT v1.5.0
- Python-dateutil v2.6.0
- Pytz v2016.10
- Six v1.10.0
- Werkzeug v0.12.1
- Yarl v0.10.2
- BeautifulSoup4 v4.6.0
- Readability-lxml v0.6.2
- Six v1.10.0
- Typing v3.6.1
- Werkzeug v0.12.1
- Yarl v0.10.2
- Git 2.13.3
- GitHub
- React.js 15.0
- Highland.js v3
- Bootstrap v3

Sprints Plan

Sprint 1

After discussion, the velocity of the team were estimated to be 100 hours total

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #1: Streams with Haskell and Conduit
- User Story #2: Linear Regression

The team members indicated their willingness to work on the following user stories.

- Alastair Paragas
 - User Story #1: Streams with Haskell and Conduit
 - User Story #2: Linear Regression
- Alberto Mizrahi
 - User Story #2: Linear Regression

Sprint 2

After discussion, the velocity of the team were estimated to be 100 hours total

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #3: Set up the basic back-end API

The team members indicated their willingness to work on the following user stories.

- Alastair Paragas
 - User Story #3: Set up the basic back-end API
- Alberto Mizrahi
 - User Story #3: Set up the basic back-end API

Sprint 3

After discussion, the velocity of the team were estimated to be 100 hours total

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #4: Understand SVMs and develop applications for cross-validation
- User Story #5: Create authentication routes
- User Story #6: Create summarization routes

The team members indicated their willingness to work on the following user stories.

- Alastair Paragas
 - User Story #5: Create authentication routes
 - User Story #6: Create summarization routes
- Alberto Mizrahi
 - User Story #4: Understand SVMs and develop applications for cross-validation
 - User Story #5: Create authentication routes
 - User Story #6: Create summarization routes

Sprint 4

After discussion, the velocity of the team were estimated to be 100 hours total

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #6: Create summarization routes
- User Story #7: Implement summarization algorithms

The team members indicated their willingness to work on the following user stories.

- Alastair Paragas
 - User Story #6: Create summarization routes
 - User Story #7: Implement summarization algorithms
- Alberto Mizrahi
 - User Story #6: Create summarization routes
 - User Story #7: Implement summarization algorithms

Sprint 5

After discussion, the velocity of the team were estimated to be 100 hours total

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #8: Design the front end and deploy the website
- User Story #9: Implement another summarization algorithm

The team members indicated their willingness to work on the following user stories.

- Alastair Paragas
 - User Story #8: Design the front end and deploy the website
 - User Story #9: Implement another summarization algorithm
- Alberto Mizrahi
 - User Story #8: Design the front end and deploy the website
 - User Story #9: Implement another summarization algorithm

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

After analyzing the requirements of the application, the team decided to utilize the Model-View-Controller (MVC) pattern as the architecture with which to design the system.



Figure 1. Diagram which illustrate the MVC architectural pattern.

In this approach, the three main actions of the system - user interaction, information processing and information storage - are designed into separate components which interact with each other externally. This allows the team to work on each component separately but concurrently, thereby increasing the work output of each team member. Furthermore, this pattern decreases the coupling and increases the cohesion between each of these components, which not only facilitates the application's overall development but also provides flexibility by allowing future developers to more easily modify the system's code.

Another architectural pattern used is the client-server approach. This consists in a set of clients sending requests to a server, which in turns satisfies them and sends their solutions through responses. Since the system had to be developed as a web application, the client-server architecture was the clear choice.

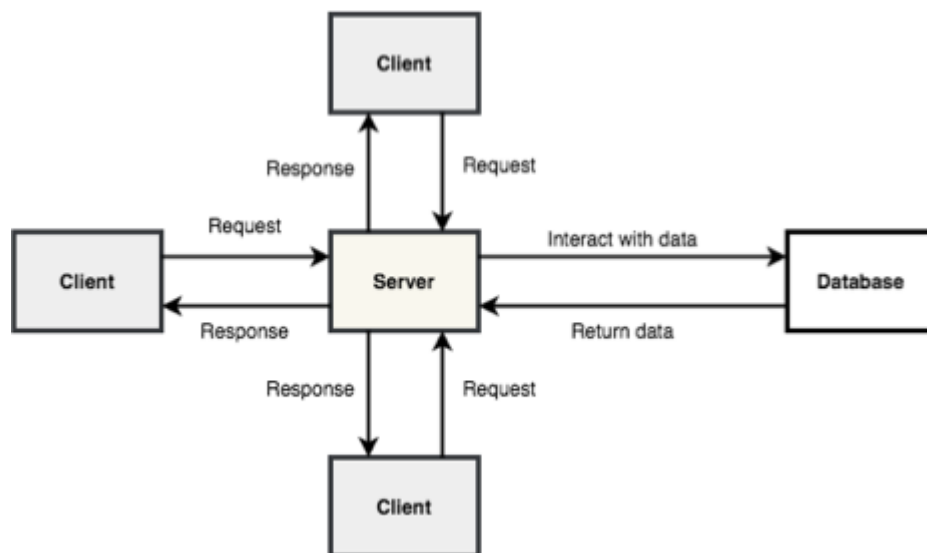


Figure 2. Diagram which illustrate the client-server architectural pattern.

System and Subsystem Decomposition

The system is composed by two major subsystems: the account processing subsystem and the summarization processing subsystem. The former is in charge of the following tasks:

- Registration of new accounts
- Allow users to log into the system
- Allow users to log out of the system

The latter subsystem covers the following actions:

- Implements various NLP summarization algorithms and allows users to choose any one of them when summarizing documents.
- Record all the documents the user has summarized.
- Allow users to retrieve the history of all the documents they have summarized.

Deployment Diagram

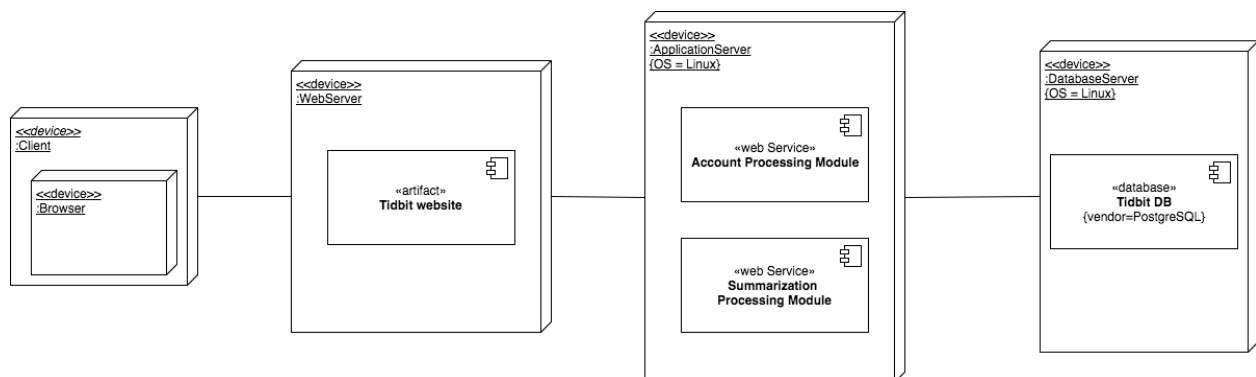


Figure 3. Deployment diagram for Tidbit.

Design Patterns

Throughout the course of the system's development, a variety of software design patterns were utilized so as to ensure that our code is efficient, reusable and flexible. The following are some examples that can be found in our codebase:

- **Object pool:** in order to avoid the constant acquisition and release of database connections, which is time consuming as well as resource-inefficient, the system maintains a pool of these connections open and ready for use.
- **Lazy initialization:** the summarization algorithms require that the document be tokenized into its corresponding sentences. If the document is large, a considerable amount of sentences must be stored in memory. So as to avoid this, these algorithms utilize a Python feature known as generators which allow us to lazily extract the sentences of the document as we need them.
- **Front controller:** since the system was designed through the MVC approach, controllers are the main points for handling web requests.
- **Modules:** a variety of modules can be found in the codebase. For instance, each summarization algorithm is implemented in its own self-contained module.
- **Command:** by implementation, Flask automatically transforms web requests into Requests objects which are independent of the client or the request type sent.
- **Iterator:** Python supports iterators as part of its language and they are used extensively throughout the system's code to iterate through lists, generators, etc.
- **Strategy:** the various summarization algorithms are contained in their respective modules and they can be easily interchanged depending on the type of algorithm the user selects.
- **Event-based asynchronous:** given that an essential component of the system is its server, which must handle multiple requests at the same, the various asynchronous features provided by Python and Flash are constantly used.

SYSTEM VALIDATION

In this section, a thorough description of the various testing suites that were developed during the course of the Tidbit's development is given. These tests allow us to state, with a high degree of confidence, that the system is valid and functioning as expected. For organizational purposes, the tests are described in the context of the user story in which they were developed. During the course of the following user stories, tests were developed:

3. Set up the basic back-end API
4. Create authentication routes
5. Create summarization routes
8. Design the front end and deploy the website

User Story #3: Set up the basic back-end API

- Test case ID: UserRepository_test_create
- Description/Summary of Test: Tests whether a User is successfully created and added to the database.
- Pre-condition: None
- Expected Results: User object is created and added to the database.
- Actual Result: User object is created and added to the database.
- Status (Fail/Pass): Pass
- Testing tools: Unittest, Asynctest

- Test case ID: UserRepository_test_delete
- Description/Summary of Test: Tests whether an existing User can be successfully deleted.
- Pre-condition: The database must already contain an existing User.
- Expected Results: User is successfully deleted.
- Actual Result: User is successfully deleted.
- Status (Fail/Pass): Pass
- Testing tools: Unittest, Asynctest

- Test case ID: UserRepository_test_update_exists
- Description/Summary of Test: Tests whether an existing User's information can be successfully updated.

- Pre-condition: The database must already contain an existing User.
 - Expected Results: User's information is successfully updated.
 - Actual Result: User's information is successfully updated.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-
- Test case ID: UserRepository_test_update_notexists
 - Description/Summary of Test: Tests whether an attempt to update the information of a non-existing User fails.
 - Pre-condition: None.
 - Expected Results: The database is not modified (since the User does not really exist).
 - Actual Result: The database is not modified.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-

User Story #4: Create authentication routes

- Test case ID: UserRepository_test_getpublic_exists
 - Description/Summary of Test: Tests that when the public information of a User is requested (which includes his or her first name, last name and username), the information is provided.
 - Pre-condition: The database must already contain an existing User.
 - Expected Results: The appropriate information is returned.
 - Actual Result: The appropriate information is returned.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-
- Test case ID: UserRepository_test_getpublic__notexists
 - Description/Summary of Test: Tests that when the public information of a non-existing User is requested (which includes his or her first name, last name and username), no information is actually returned.

- Pre-condition: None.
 - Expected Results: No information is returned.
 - Actual Result: No information is returned.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asyncctest
-
- Test case ID: UserRepository_test_getprivate
 - Description/Summary of Test: Tests that when the private information of a non-existing User is requested (which includes his or her first name, last name, username, password as well as his or her history of summarized documents), the information is returned as long as the entity requesting this information is authorized to see it.
 - Pre-condition: The database must already contain an existing User.
 - Expected Results: The User's private information is returned as long as the entity requesting it is authorized to do so.
 - Actual Result: The User's private information is returned as long as the entity requesting it is authorized to do so.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asyncctest
-
- Test case ID: UserRepository_test_getbycredentials_exists
 - Description/Summary of Test: Tests that when the credential informations of a User is requested (which includes his or her user ID and username), the information is returned as long as the entity requesting this information is authorized to see it.
 - Pre-condition: The database must already contain an existing User.
 - Expected Results: The User's credential information is returned as long as the entity requesting it is authorized to do so.
 - Actual Result: The User's credential information is returned as long as the entity requesting it is authorized to do so.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asyncctest
-
- Test case ID: UserRepository_test_getbycredentials_notexists
 - Description/Summary of Test: Tests that when the credential informations of a non-existing User is requested (which includes his or her user ID and username), nothing is returned.
 - Pre-condition: None.

- Expected Results: No information is returned.
 - Actual Result: No information is returned.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-

User Story #5: Create summarization routes

- Test case ID: DocumentRepository_test_create
 - Description/Summary of Test: Tests whether a Document is successfully created and added to the database when it is summarized.
 - Pre-condition: Must have an existing User in the database.
 - Expected Results: Document object is created and added to the database.
 - Actual Result: Document object is created and added to the database.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-
- Test case ID: DocumentRepository_test_getby_id
 - Description/Summary of Test: Tests whether an existing Document in the database is successfully returned when its ID is given. If the Document does not exist, nothing is returned.
 - Pre-condition: Must have an existing User in the database and a Document associated to that User.
 - Expected Results: Document is successfully returned if it exists and nothing is returned if it does not exist.
 - Actual Result: Document is successfully returned if it exists and nothing is returned if it does not exist.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-
- Test case ID: DocumentRepository_test_getby_url

- Description/Summary of Test: Tests whether an existing Document in the database is successfully returned when its corresponding URL is given. If the Document does not exist, nothing is returned.
 - Pre-condition: Must have an existing User in the database and a Document associated to that User.
 - Expected Results: Document is successfully returned if it exists and nothing is returned if it does not exist.
 - Actual Result: Document is successfully returned if it exists and nothing is returned if it does not exist.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-
- Test case ID: HistoryRepository_test_create
 - Description/Summary of Test: Tests whether a History object is successfully created and added to the database when a Document is summarized.
 - Pre-condition: Must have an existing User in the database.
 - Expected Results: History object is created and added to the database.
 - Actual Result: History object is created and added to the database.
 - Status (Fail/Pass): Pass
 - Testing tools: Unittest, Asynctest
-

User Story #8: Design the front end and deploy the website

- Test case ID: Selenium_test_user_registration
- Description/Summary of Test: Checks whether a User is created and logged in by filling the user registration form.
- Pre-condition: None.
- Expected Results: User is registered and then logged in.
- Actual Result: User is registered and then logged in.
- Status (Fail/Pass): Pass

- Testing tools: Selenium
- Test case ID: Selenium_test_user_login
- Description/Summary of Test: Checks whether an existing User can log in by filling the user log in form.
- Pre-condition: The system must have an a User already registered.
- Expected Results: User is logged in.
- Actual Result: User is logged in.
- Status (Fail/Pass): Pass
- Testing tools: Selenium
- Test case ID: Selenium_test_user_logout
- Description/Summary of Test: Checks that a logged in User can log out by clicking the “Logout” link.
- Pre-condition: The system must have an a User already registered and the User must be logged in.
- Expected Results: User is logged out.
- Actual Result: User is logged out.
- Status (Fail/Pass): Pass
- Testing tools: Selenium
- Test case ID: Selenium_test_summarization_not_logged_in
- Description/Summary of Test: Checks that if a User attempts to summarize an article without logging in, he or she obtains an error message.
- Pre-condition: None.
- Expected Results: Error message is shown.
- Actual Result: Error message is shown.
- Status (Fail/Pass): Pass
- Testing tools: Selenium
- Test case ID: Selenium_test_summarization_logged_in
- Description/Summary of Test: Checks that a logged in User can summarize an article by inputting its URL, algorithm type and clicking “Summarize”
- Pre-condition: The system must have an a User already registered and the User must be logged in.
- Expected Results: Article is summarized.
- Actual Result: Article is summarized.
- Status (Fail/Pass): Pass

- Testing tools: Selenium
 - Test case ID: Selenium_test_view_history
 - Description/Summary of Test: Checks that after summarizing an article, it is added to the User's history and it is shown in the View History page.
 - Pre-condition: The system must have an a User already registered and the User must be logged in.
 - Expected Results: Summarized article is added to the User's history.
 - Actual Result: Summarized article is added to the User's history.
 - Status (Fail/Pass): Pass
 - Testing tools: Selenium
-

After each of the user stories were implement and unit tested separately, integration tests were conducted to ensure that the modifications brought about by the new user story did not have an adverse effect on the components of the system that were already known and tested to be working correctly. Hence, after each new user story, all the testing suites of the various subsystems were executed and it was made sure they all finished successfully. In this way, the code implemented in every user story was seamlessly integrated into the existing system.

GLOSSARY

Machine Learning: A field of computer science that consists in giving computers the ability to solve problems without being explicitly and directly programmed to do so.

Natural Language Processing (NLP): A field of artificial intelligence, computational linguistics and computer science that studies different methods by which computers can be “taught” to understand human language.

Summarization: The process by which a set of coherent and informative textual data is shortened while still preserving its essential ideas.

APPENDIX

Appendix A - UML Diagrams

Static UML Diagrams

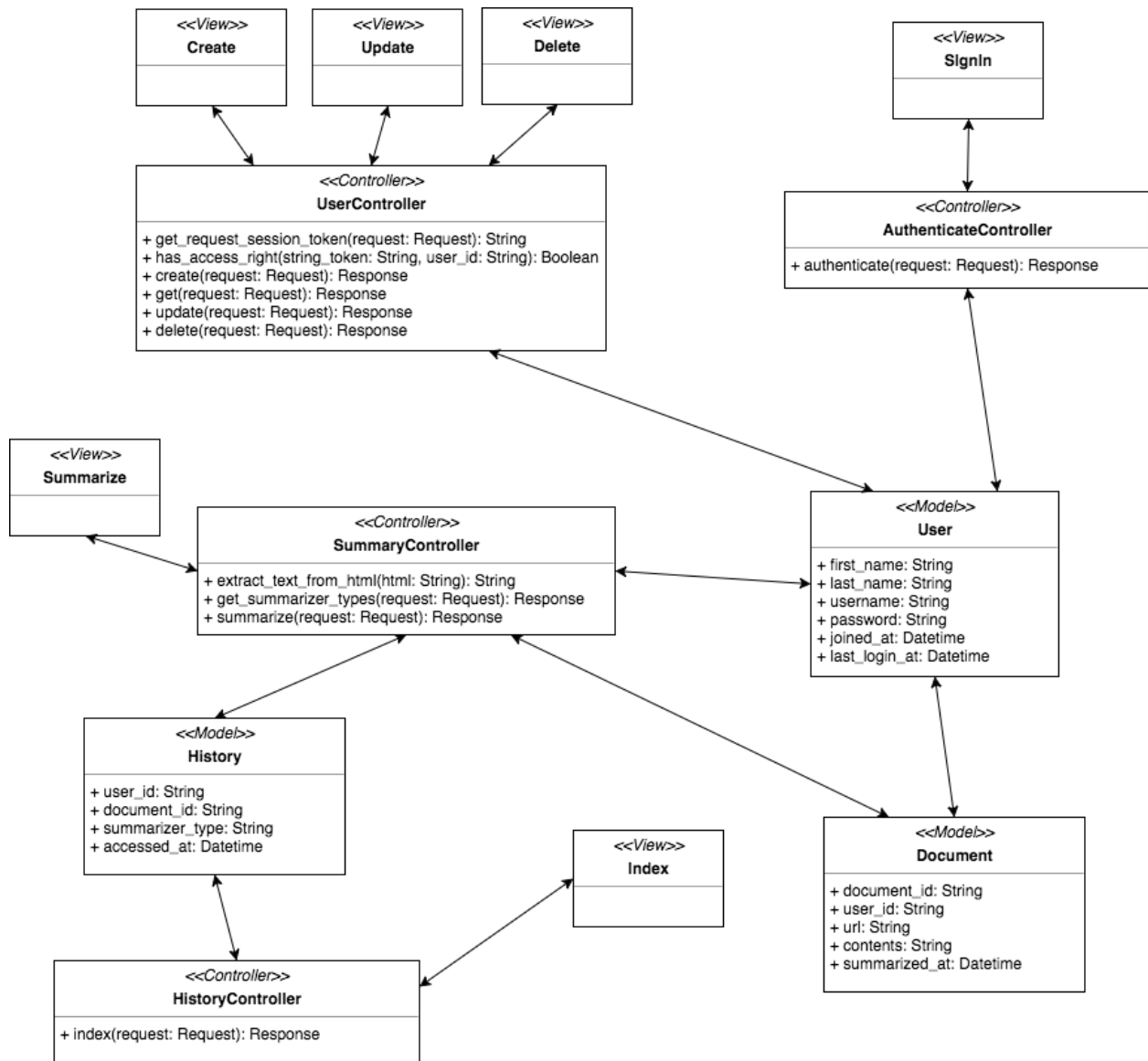


Figure A.1. Object design of the entire system.

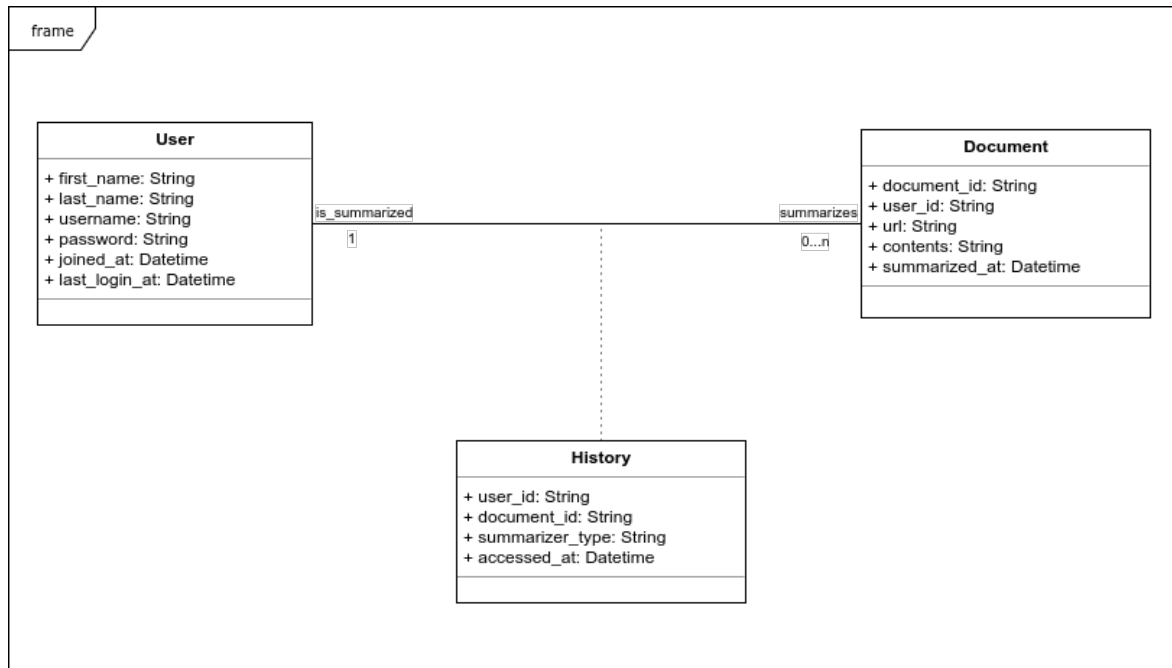


Figure A.2. Class diagram which shows an abstract view of the relationships between the entity objects of the application.

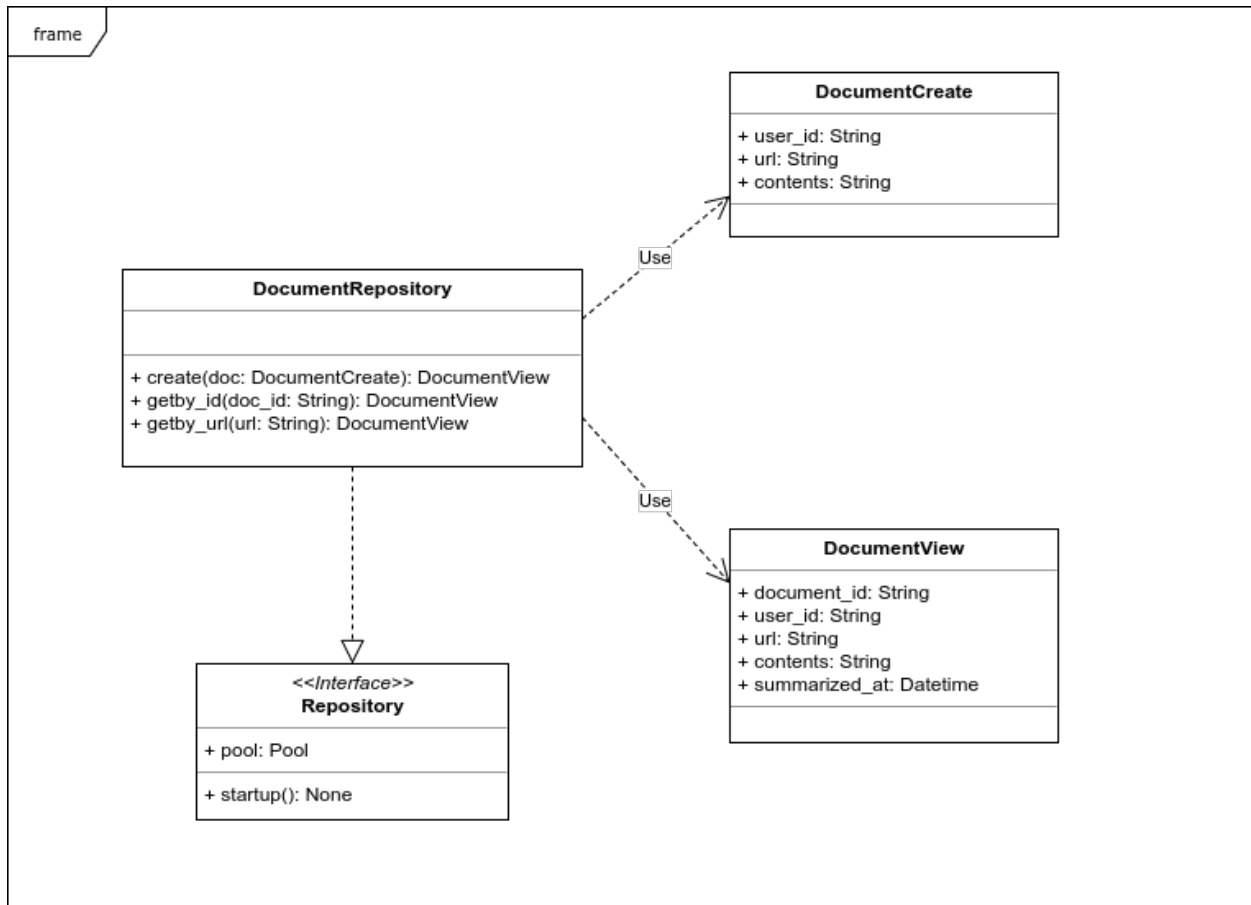


Figure A.3. Class diagram which shows all the various interactions between the classes, modules and libraries of the application as they relate to the Document entity.

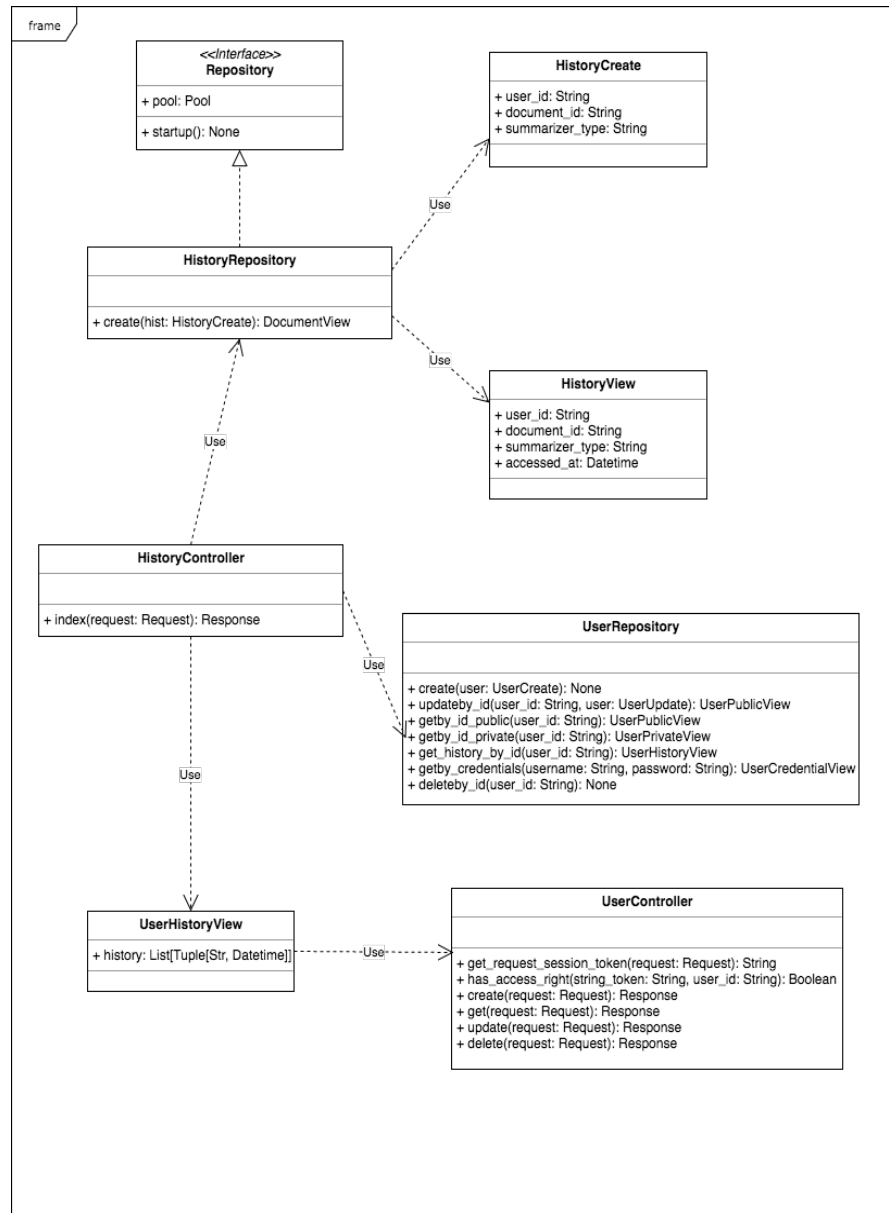


Figure A.4. Class diagram which shows all the various interactions between the classes, modules and libraries of the application as they relate to the History entity.

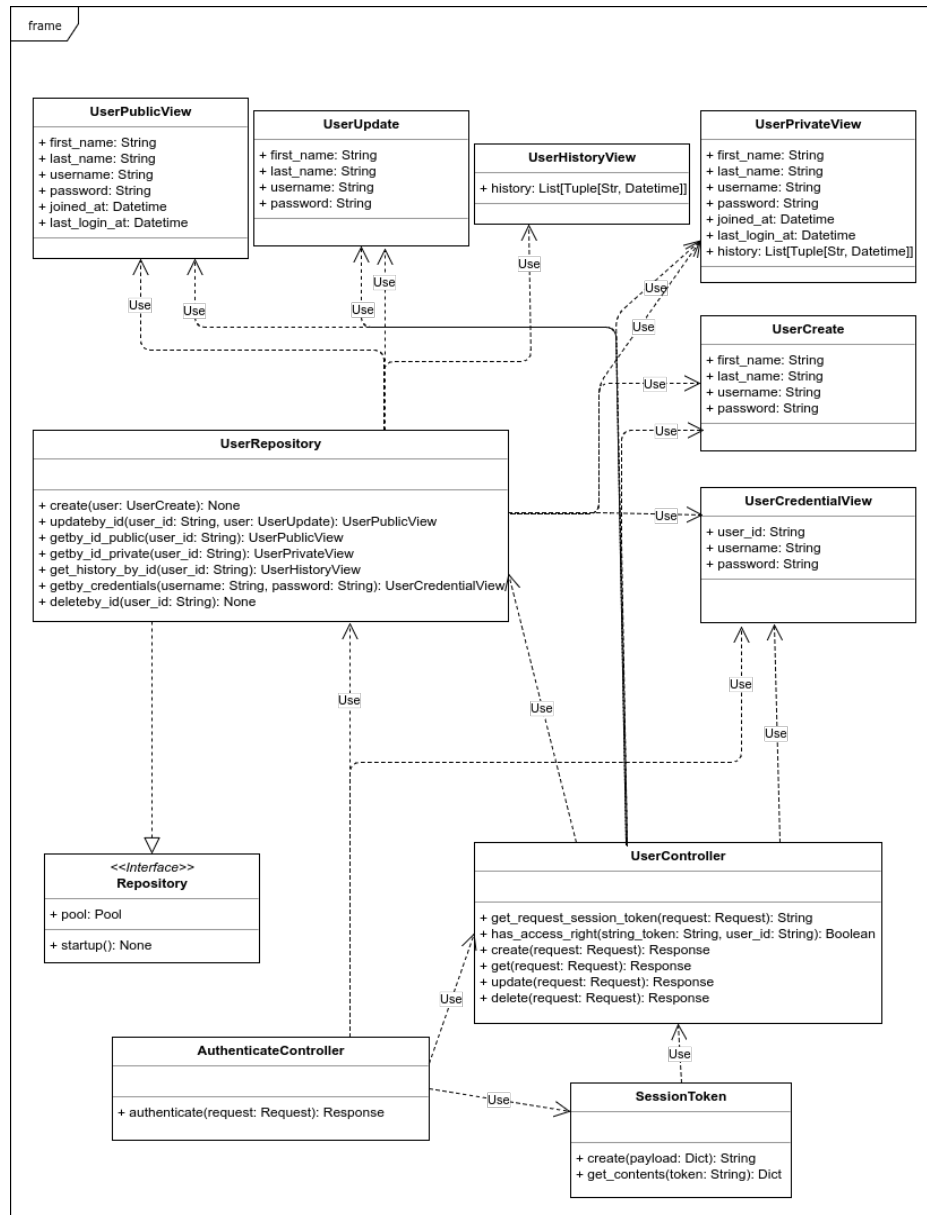


Figure A.5. Class diagrams which shows all the various interactions between the classes, modules and libraries of the application as they relate to the User entity.

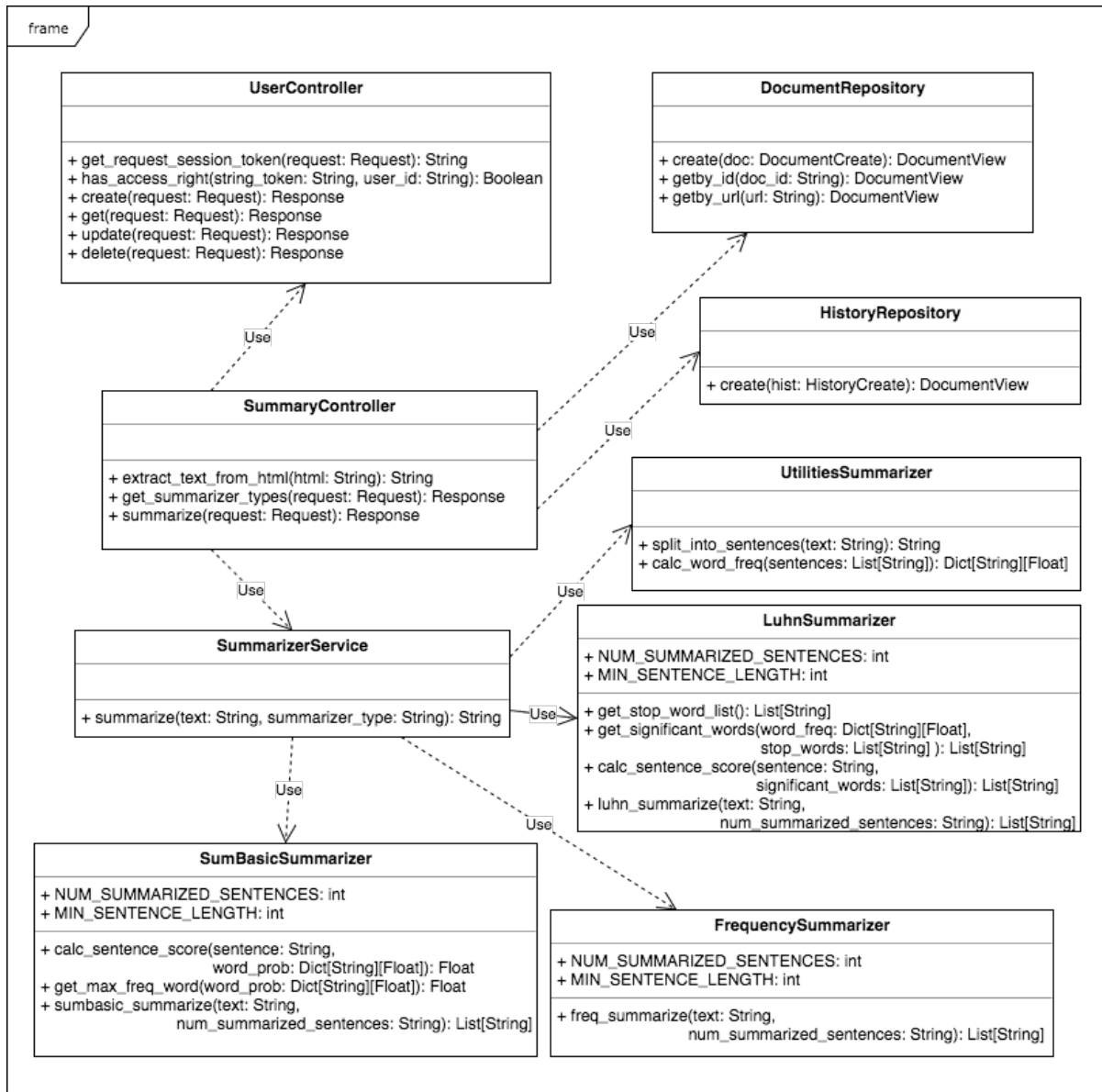


Figure A.6. Class diagram which shows all the various interactions between the classes, modules and libraries of the application when an User requests the summarization of a Document.

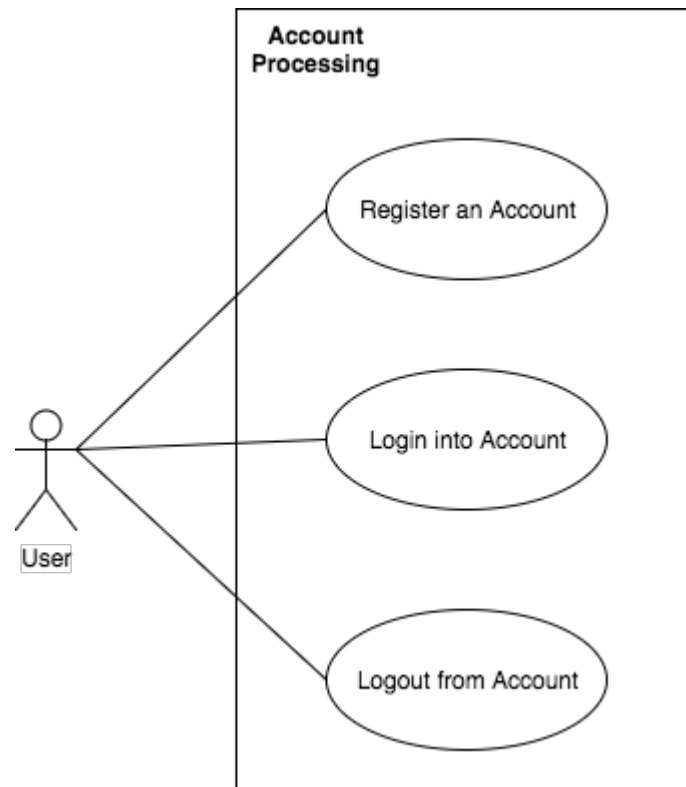


Figure A.7. Diagram that illustrates use cases related to account registration and login.

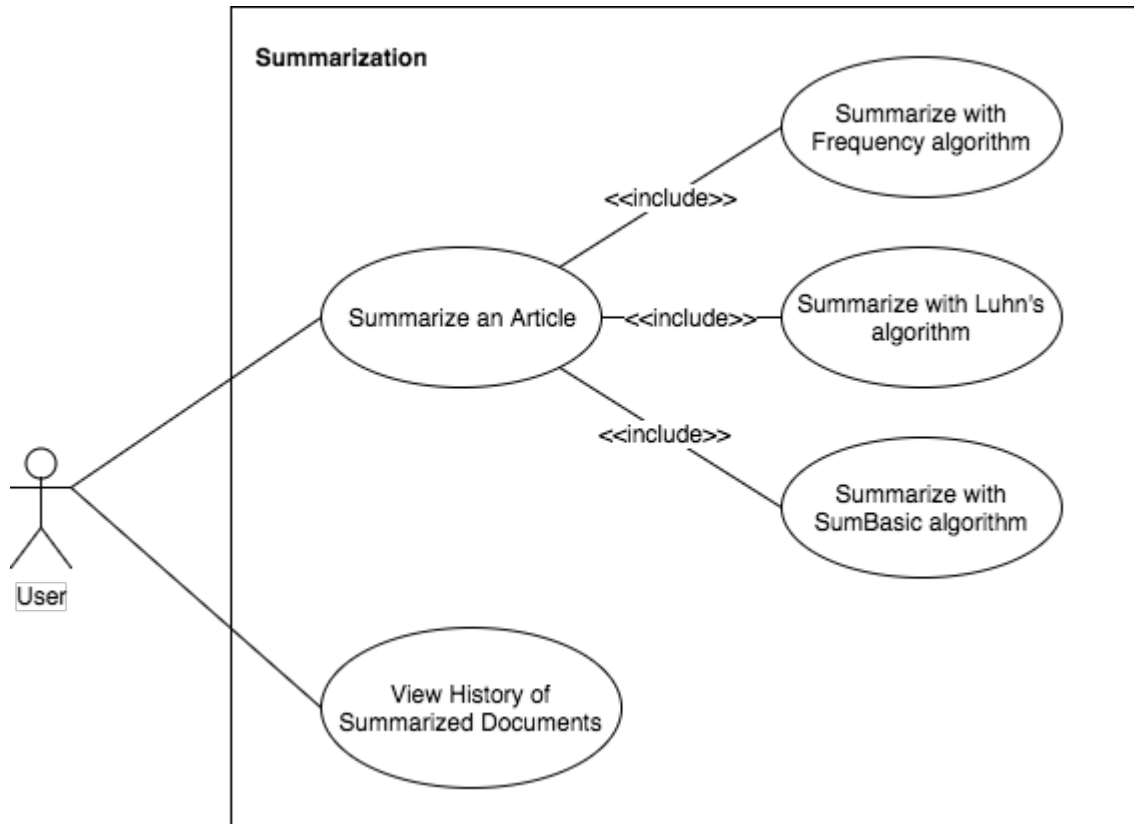
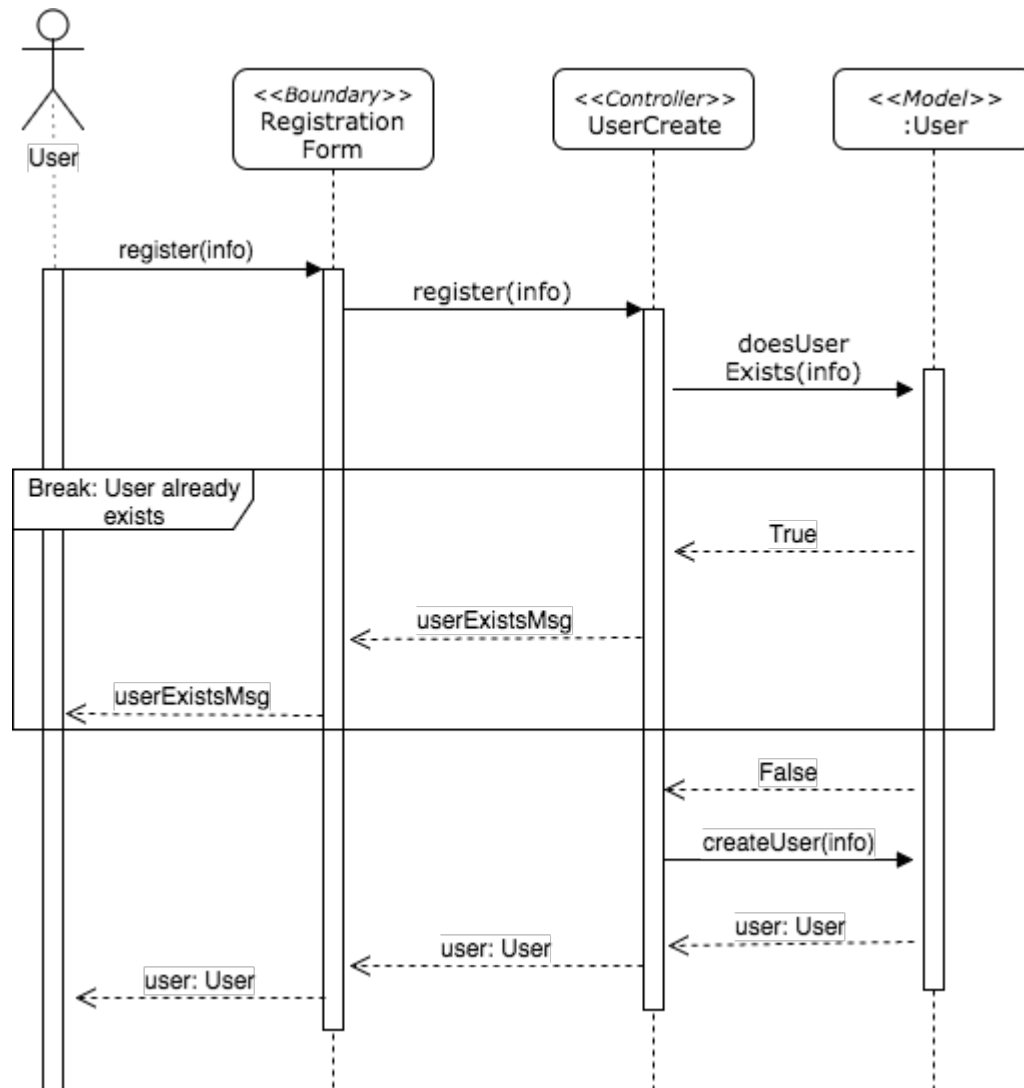


Figure A.8. Diagram that illustrates use cases related to the summarization of articles.

Dynamic UML Diagrams**Figure A.9.** Sequence diagram for User registration.

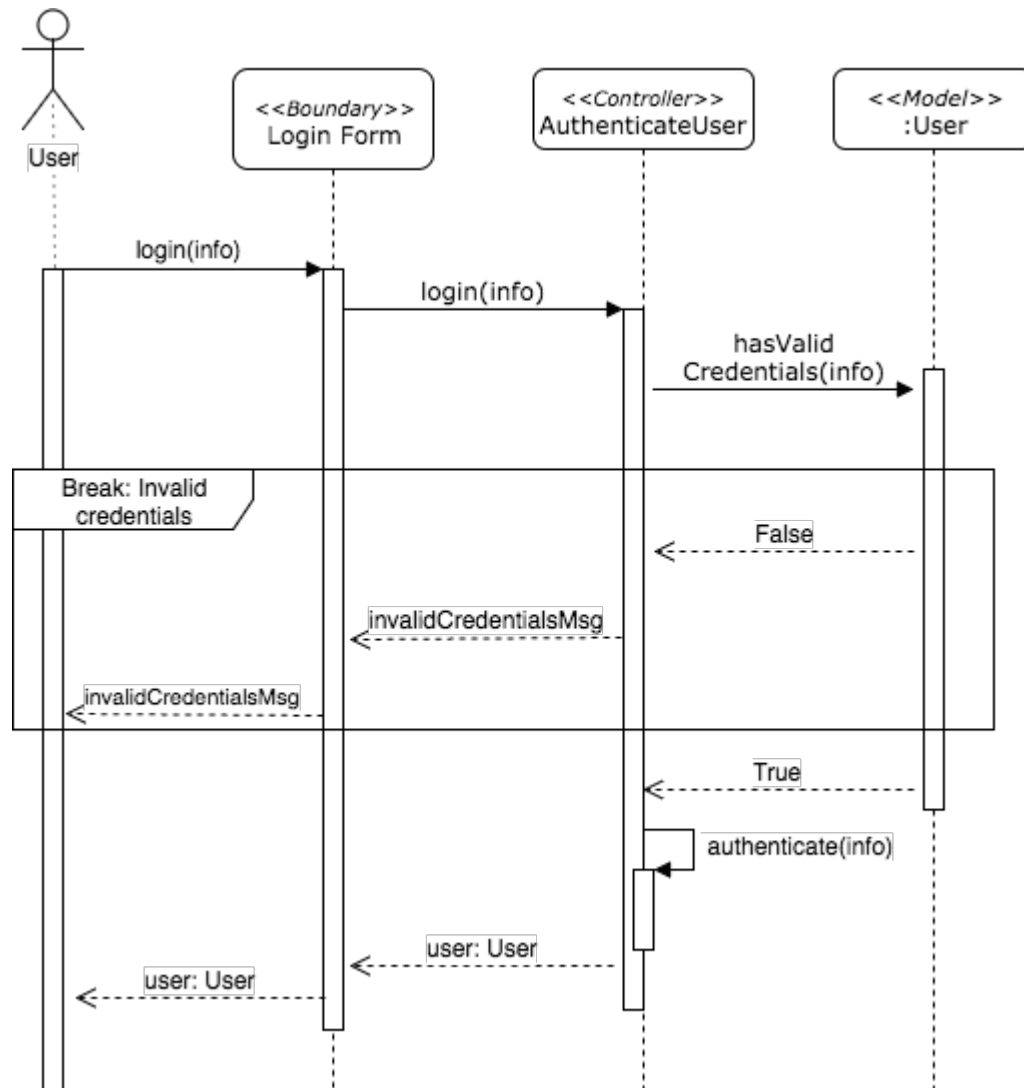


Figure A.10. Sequence diagram for User login.

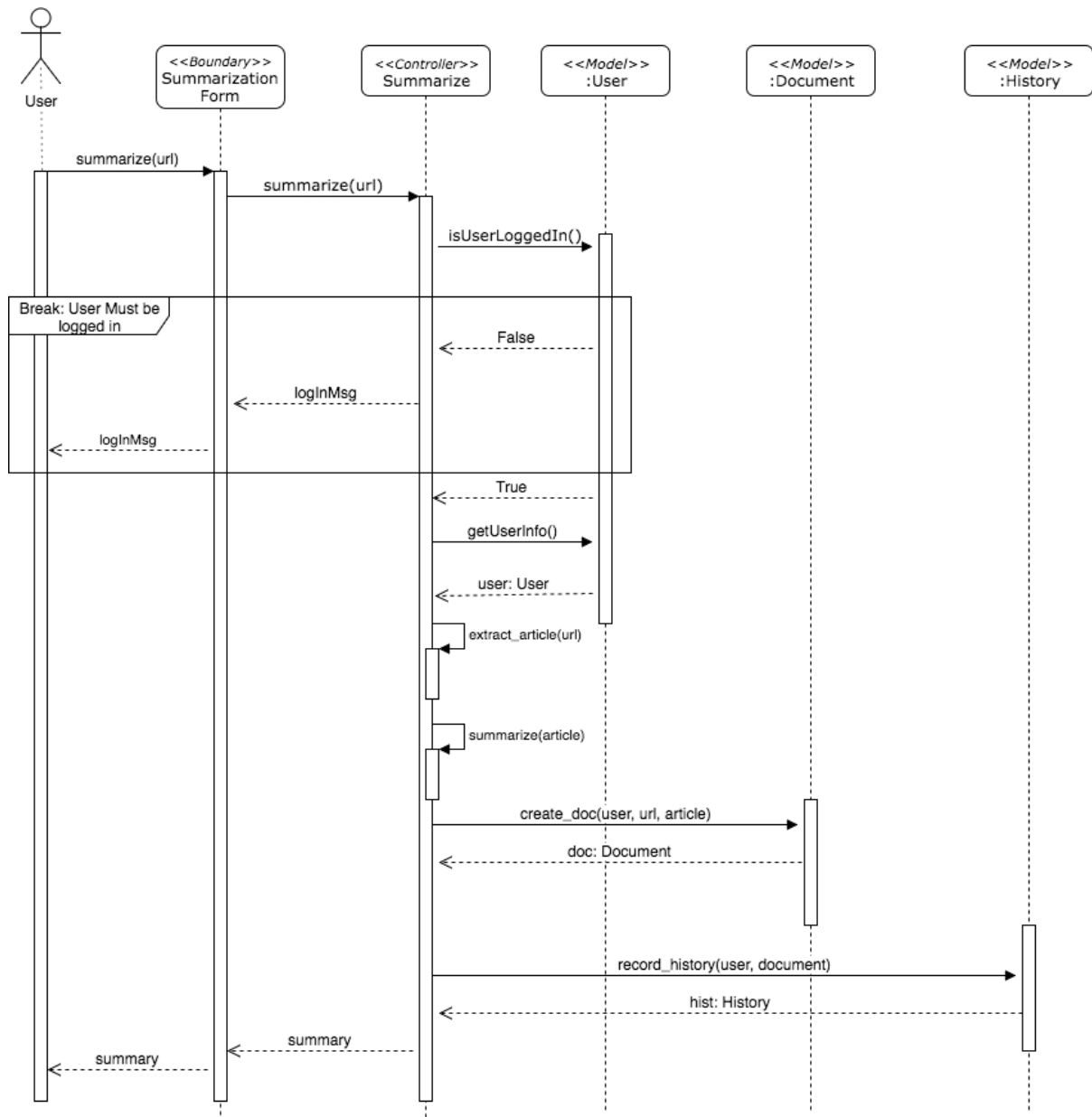


Figure A.11. Sequence diagram for the summarization of an article.

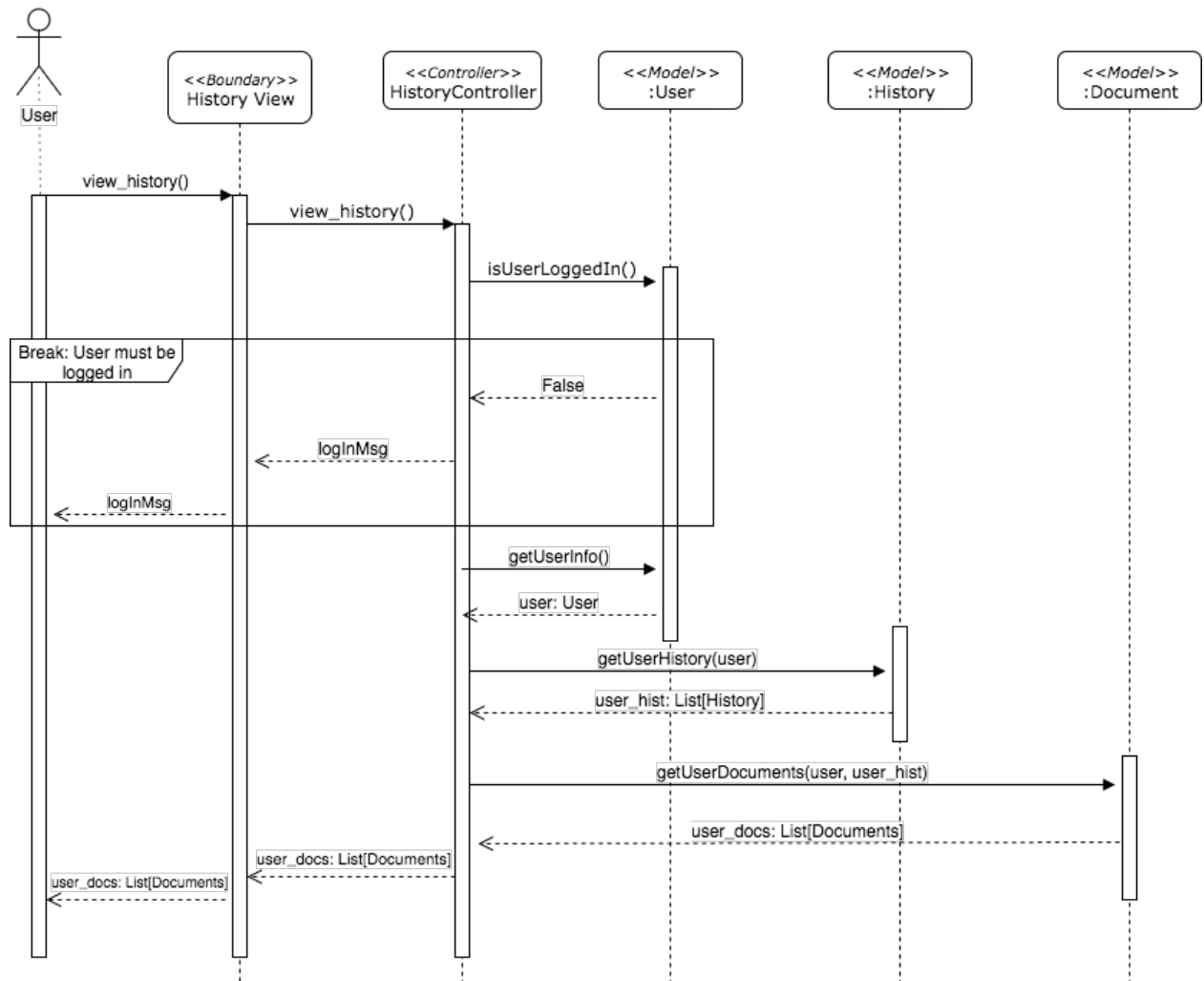
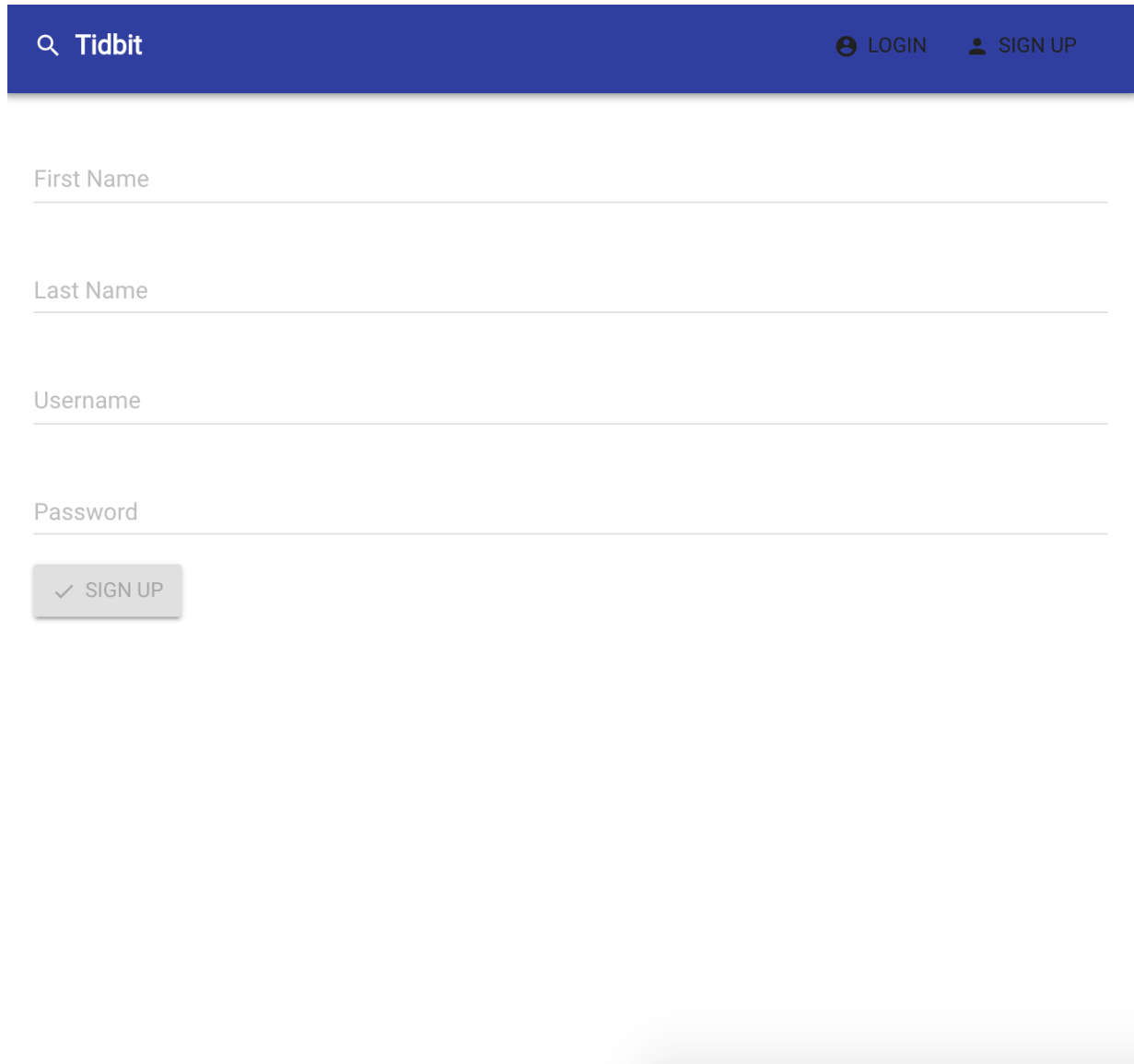


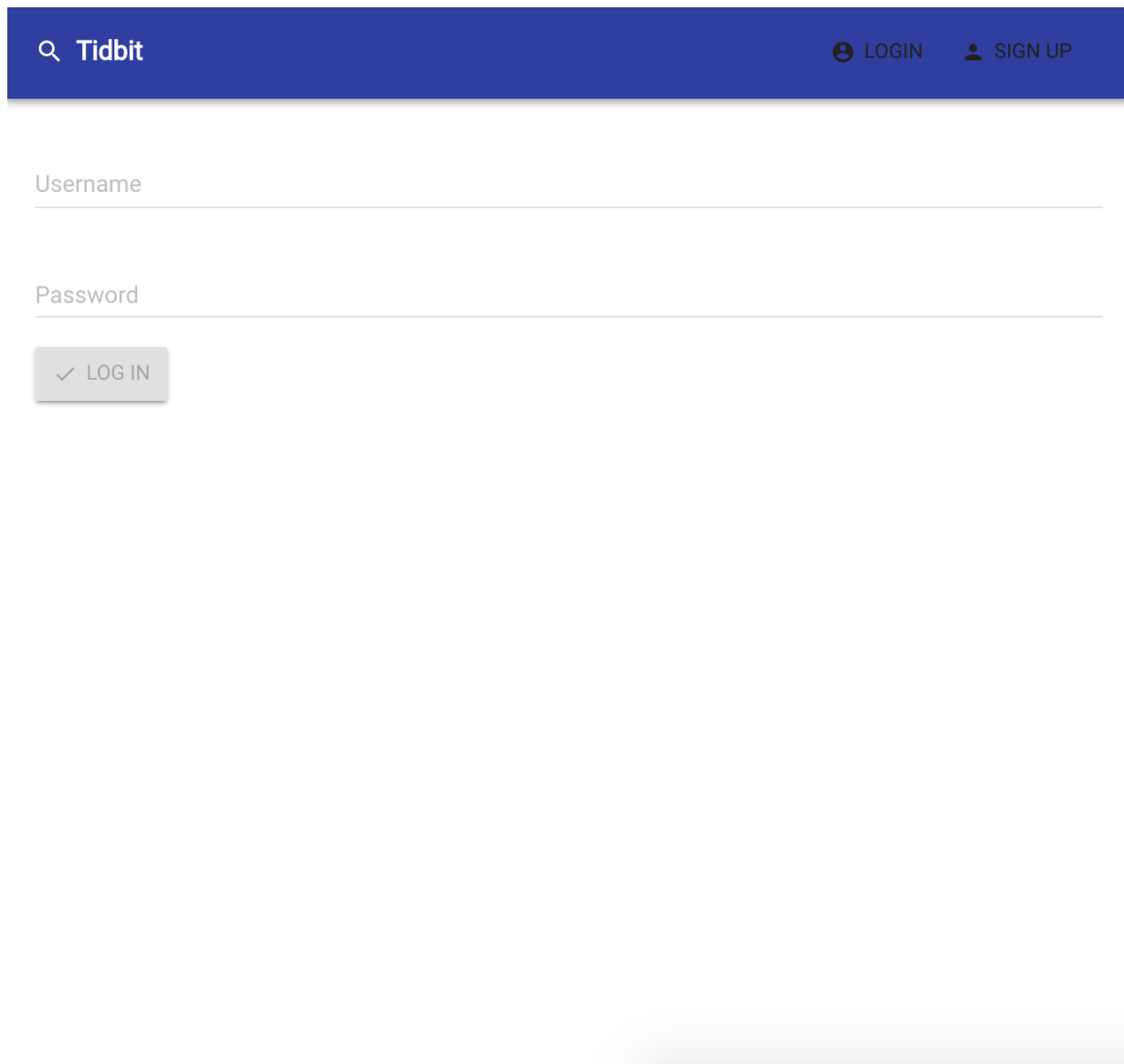
Figure A.12. Sequence diagram for the summarization of an article.

Appendix B - User Interface Design



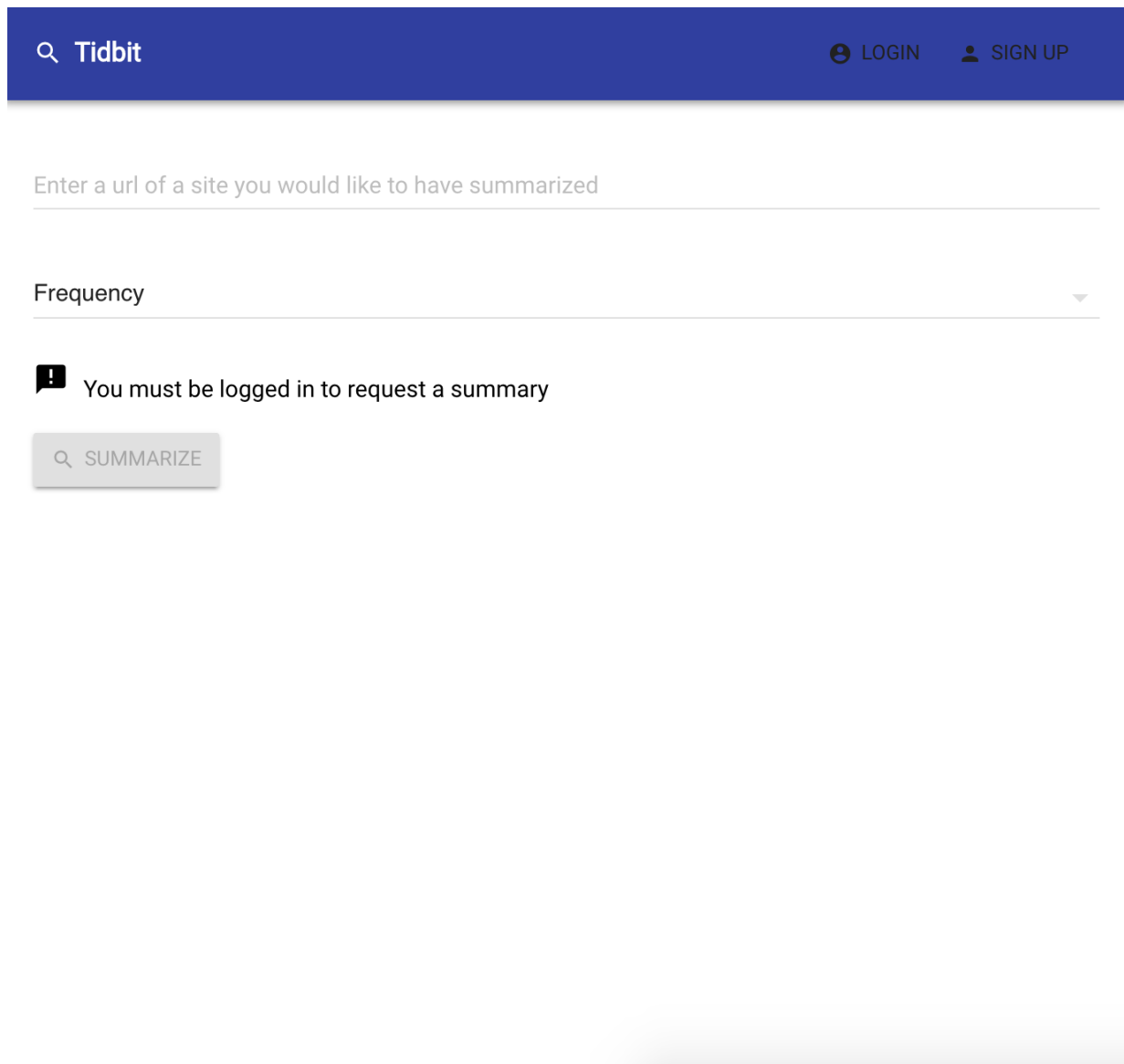
The image shows a user registration form for a website called 'Tidbit'. At the top is a dark blue header bar with the 'Tidbit' logo on the left and 'LOGIN' and 'SIGN UP' links on the right. Below the header, the form consists of four text input fields labeled 'First Name', 'Last Name', 'Username', and 'Password'. At the bottom of the form is a grey button with a checkmark icon and the text 'SIGN UP'.

Figure B.1. The user registration form.



The image shows a user login form for a service called 'Tidbit'. At the top, there is a dark blue header bar. On the left side of this bar is a magnifying glass icon followed by the text 'Tidbit'. On the right side of the bar are two links: 'LOGIN' with a person icon and 'SIGN UP' with a person icon. Below the header bar, the form is set against a light gray background. It features two input fields: the first is labeled 'Username' and the second is labeled 'Password'. Below the 'Password' field is a gray button with a checkmark icon and the text 'LOG IN'.

Figure B.2. The user login form.



The screenshot shows the Tidbit web application interface. At the top is a dark blue header bar with the Tidbit logo on the left and 'LOGIN' and 'SIGN UP' links on the right. Below the header is a light gray input field with the placeholder text 'Enter a url of a site you would like to have summarized'. Underneath this is a 'Frequency' dropdown menu. A red error message with an exclamation mark icon states: 'You must be logged in to request a summary'. At the bottom of the form is a gray button with a magnifying glass icon and the text 'SUMMARIZE'.

Figure B.3. The error message that appears when a User tries to summarize an article without first login in.

Tidbit

×

LOGOUT

Enter a url of a site you would like to have summarized

http://edition.cnn.com/2017/06/12/asia/philippines-isis-marawi-duterte/index.html

Frequency

Q SUMMARIZE

Success

Algorithm

Summarization algorithm used

FREQUENCY

Summary

Summary obtained

As the fighting wears on, deadline after deadline set by the Philippines government for the end of the conflict has been missed, including a pledge to finish the fighting by Monday, the country's Independence Day. About 1,000 civilians remain trapped in the war-torn city, according to the city's mayor". The fighting has so far claimed the lives of 58 government troops and over 100 civilians. The siege on Marawi unfolded as Muslims worldwide began to mark the holy month of Ramadan. The number of troops there ranges between 50 to 100 at any given time, the Pentagon said.

Figure B.4. An example of an article that is summarized by the system.



Figure B.5. A User's summarization history.

Appendix C - Sprint Review Reports

Sprint 1

Attendees: Alastair Paragas, Alberto Mizrahi, Nagarajan Prabakar

Start time: 05/30/2017, 10:00 AM

End time: 05/30/2017, 11:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #1: Streams with Haskell and Conduit
- User Story #2: Linear Regression

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- User Story #4: Understand SVMs and develop applications for cross-validation

Sprint 2

Attendees: Alastair Paragas, Alberto Mizrahi, Nagarajan Prabakar

Start time: 06/9/2017, 3:00 PM

End time: 06/9/2017, 4:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #3: Set up the basic back-end API

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- User Story #4: Understand SVMs and develop applications for cross-validation

Sprint 3

Attendees: Alastair Paragas, Alberto Mizrahi, Nagarajan Prabakar

Start time: 06/25/2017, 4:00 PM

End time: 06/25/2017, 5:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #4: Understand SVMs and develop applications for cross-validation
- User Story #5: Create authentication routes

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- User Story #6: Create summarization routes

Sprint 4

Attendees: Alastair Paragas, Alberto Mizrahi, Nagarajan Prabakar

Start time: 07/09/2017, 12:00 PM

End time: 07/09/2017, 1:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #6: Create summarization routes
- User Story #7: Implement summarization algorithms

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- None

Sprint 5

Attendees: Alastair Paragas, Alberto Mizrahi, Nagarajan Prabakar

Start time: 07/14/2017, 12:00 PM

End time: 07/14/2017, 1:00 PM

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #699 - Design the front end and deploy the website
- User Story #700 - Implement another summarization algorithm

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- None

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

User Manual

The following is the homepage of the application:

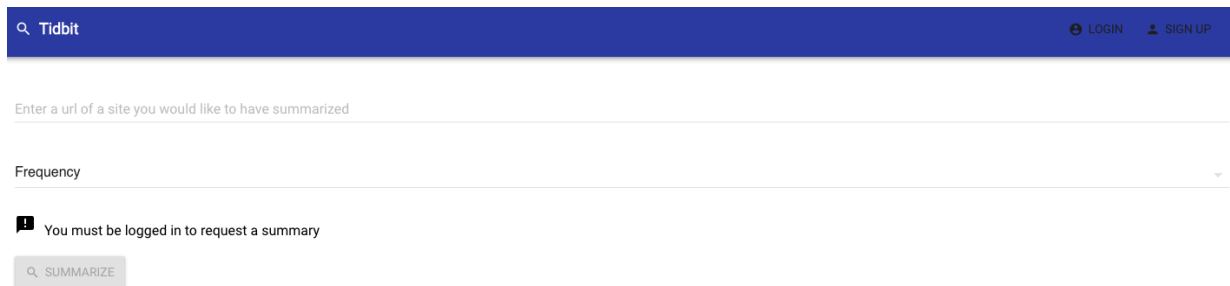
A screenshot of the Tidbit application homepage. The top navigation bar is dark blue with a search icon and the text 'Tidbit' on the left, and 'LOGIN' and 'SIGN UP' links on the right. Below the navigation bar is a large white input field with the placeholder text 'Enter a url of a site you would like to have summarized'. Underneath this is a 'Frequency' dropdown menu. A message in a small box states 'You must be logged in to request a summary'. At the bottom is a 'SUMMARIZE' button.

Figure D.1. The homepage of Tidbit.

Registering an account

To register for an account, in the homepage, click the “Sign Up” button at the top right corner of the menu bar, as shown below in the red box:

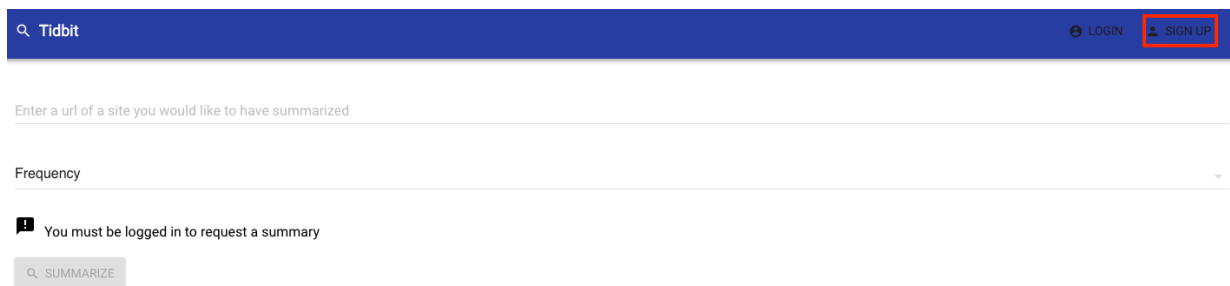
A screenshot of the Tidbit application homepage, identical to Figure D.1, but with a red rectangular box highlighting the 'SIGN UP' link in the top right corner of the dark blue navigation bar.

Figure D.2. Showing the Sign Up button.

After clicking the button, you will be taken to the following registration form:

A registration form for the Tidbit application. The form is set against a dark blue header bar. The header bar contains a search icon and the text 'Tidbit' on the left, and 'LOGIN' and 'SIGN UP' links on the right. The form fields are: 'First Name', 'Last Name', 'Username', and 'Password', each with a light gray placeholder text. Below the 'Password' field is a 'SIGN UP' button with a checkmark icon.

Figure D.3. The registration form.

After filling in the information, click the “Sign Up” button and your account will be created.

Login into your account

To log in into your account, click the “Login” button at the top right corner of the menu bar, as shown below in the red rectangle:

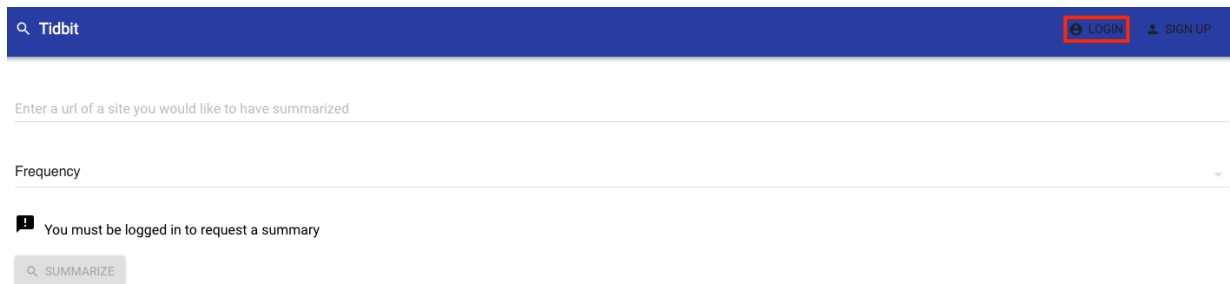
A login form for the Tidbit application. The form is set against a dark blue header bar. The header bar contains a search icon and the text 'Tidbit' on the left, and a 'LOGIN' button (highlighted with a red rectangle) and a 'SIGN UP' link on the right. The form fields are: 'Enter a url of a site you would like to have summarized' and 'Frequency'. Below the 'Frequency' field is a message: 'You must be logged in to request a summary'. At the bottom is a 'SUMMARIZE' button with a search icon.

Figure D.4. Showing the Login button.

You will then be shown the following form:

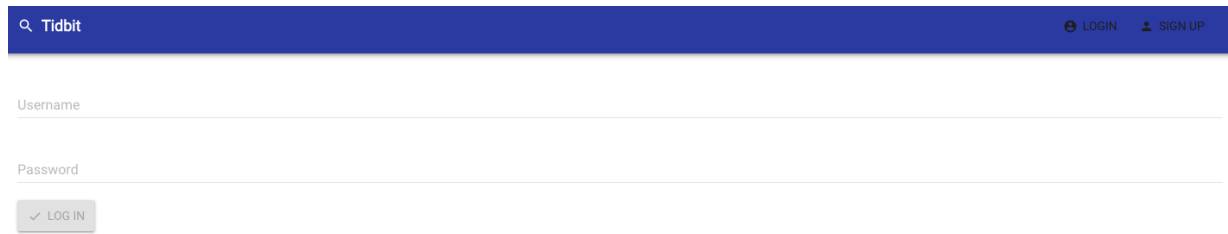
The image shows the login interface of the Tidbit application. At the top is a dark blue header bar with the 'Tidbit' logo on the left and 'LOGIN' and 'SIGN UP' links on the right. Below the header, there are two input fields: 'Username' and 'Password'. A 'LOG IN' button with a checkmark icon is positioned below the password field.

Figure D.5. The Login form.

After filling in the information, click the “Log In” button and you will be logged in.

Summarizing a document

After logging in into your account, the following form will be shown:

The image shows the summarization interface of the Tidbit application. It features a dark blue header bar with the 'Tidbit' logo on the left and 'HISTORY' and 'LOGOUT' links on the right. Below the header, there is a large text input field with the placeholder text 'Enter a url of a site you would like to have summarized'. Below this field is a 'Frequency' dropdown menu. At the bottom left, there is a red 'SUMMARIZE' button with a magnifying glass icon.

Figure D.6. The Summarization form.

In the first field, input the URL of the document or article you want to summarize. In the second field, choose the summarization algorithm you wish to utilize. Afterwards, click the “Summarize” button. The summary will then be shown:

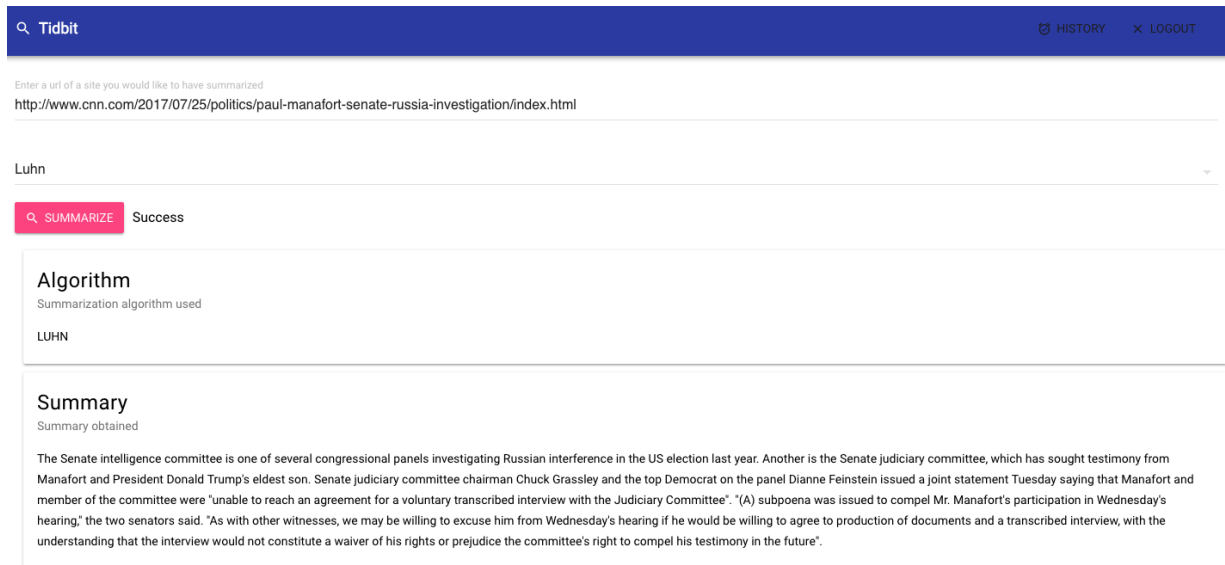


Figure D.7. An example of a summarization of an article.

Viewing your summarization history

To view the history of the documents you have summarized, log in into your account and then click the “History” button on the top right corner of the menu bar, as shown below in the red box:



Figure D.8. Showing the History button.

After clicking it, you will be shown all the documents you have summarized as well the times when you summarized them:



Figure D.9. Showing the summarization history of an account.

Log out of your account

To log out of your account, click the “Logout” button on the top right corner of the menu bar, as shown below in the red box:

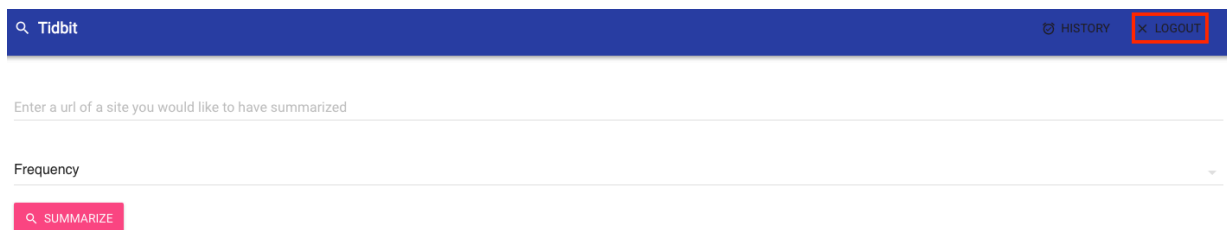


Figure D.10. The Logout button.

Sprint Retrospective Reports

Sprint 1

Attendees: Alastair Paragas, Alberto Mizrahi

Start time: 05/30/2017, 10:00 AM

End time: 05/30/2017, 11:00 AM

What went wrong?

- Did we do a good job estimating our team's velocity?
 - Alastair: We did a good job estimating the team's velocity.
 - Alberto: Yes, the estimate was correct because we were able to finish all of the tasks in the selected user stories on time.
- Did we do a good job estimating the points (time required) for each user story?
 - Alastair: Yes, although the readings are 15-30 pages (for each lecture notes), it did require concentration and focus to understand the proofs and concepts behind some of the algorithms.
 - Alberto: For the most part, yes. Reading each lecture note takes a long time, but we also have to account for the fact that in order to understand some of the ideas, we have to review some concepts (e.g. Probability and Statistics, Linear Algebra, etc.) so that we can properly understand the content of the notes. This review, of course, also takes more time.
- Did each team member work as scheduled?
 - Alberto: Yes, every team member stuck to the schedule and managed to complete all of his or her tasks in time.
 - Alastair: Yes, everyone worked as scheduled.

What went right?

- Alastair: Since this sprint went mostly on research, the team mostly focused on doing readings and doing research-based implementation and everyone did a good job about it (with good proof that the research work was done). I have not finished my Lecture Notes #2, but I shall finish it for the second sprint.
- Alberto: Our goal for this sprint, which was to begin research about machine learning, went as planned. We were all able to study the research material that was assigned for this sprint.

How to address the issues in the next sprint?

- How to improve the process?
 - Alastair: Be more disciplined about doing our Sprint documentation on time (Sprint Review and Retrospective right during the last day of the Sprint, Sprint

Planning during the first day of the Spring) and setting up non-implementation logistics more carefully.

- Alberto: Be more punctual when it comes to the meetings. We should also be more responsible when it comes to updating Mingle as details about the user stories and tasks change.
- How to improve the product?
 - Alastair: Start working on bits of actual implementation for the project.
 - Alberto: Continue doing research and acquiring enough of an intellectual foundation so that we can conduct product development properly once the time comes.

Sprint 2

Attendees: Alastair Paragas, Alberto Mizrahi

Start time: 06/09/2017, 3PM

End time: 06/09/2017, 4PM

What went wrong?

- Did we do a good job estimating our team's velocity?
 - Alberto Mizrahi: No we did not. I was not able to finish the task of reading lecture notes #5 as well as the task of developing an application that implements cross-validation for User Story #669.
 - Alastair Paragas: I wasn't able to finish some of my readings for the User Story #669 as I didn't finish reading Notes #3.
- Did we do a good job estimating the points (time required) for each user story?
 - Alberto Mizrahi: No we did not. Since we were not able to finish all the tasks, it is clear that we underestimated the points for the user story.
 - Alastair Paragas: We didn't do a good job as we underestimated the points for the user stories
- Did each team member work as scheduled?

- Alberto Mizrahi: Every team member worked the hours required of him. The issue that is that we underestimated the time it would take to do the user stories.
- Alastair Paragas: Everyone worked on the team velocity of 100 hours but we underestimated the points required for our user stories.

What went right?

- Alastair: Documentation and a basic foundation of the backend was successfully completed
- Alberto: The basic back-end functionality was successfully completed. We were also able to read most of the lecture notes that we had to.

How to address the issues in the next sprint?

- How to improve the process?
 - Alastair: Don't underestimate points for user stories
 - Alberto: Be more careful when estimating the time required to complete a user story or task.
- How to improve the product?
 - Alastair: Be more aggressive in implementation of pieces of the product
 - Alberto: Begin the implementation of functionality as soon as possible.

Sprint 3

Attendees: Alastair Paragas, Alberto Mizrahi

Start time: 06/25/2017, 4PM

End time: 06/25/2017, 5PM

What went wrong?

- Did we do a good job estimating our team's velocity?
 - Alberto Mizrahi: No we did not. We were not able to finish User Story User Story #690 - Create summarization routes.

- Alastair Paragas: No, we did not. We were not able to finish User Story #690 - Create Summarization routes because of backtracking and working on bugs that emanated from the previous sprint.
- Did we do a good job estimating the points (time required) for each user story?
 - Alberto Mizrahi: No we did not. We were not able to finish all the planned stories, thus we underestimated the points for them.
 - Alastair Paragas: We were not able to cover all our assigned user stories, especially the ones dealing with creating summarization routes.
- Did each team member work as scheduled?
 - Alberto Mizrahi: Every team member worked the hours required of him. Our only problem was we underestimated the time it would take to do the user stories.
 - Alastair Paragas: Alberto and I worked as diligently as we could to try to not fall behind on our user stories, but we severely underestimated the time.

What went right?

- Alastair: I was able to finish Dockerizing and doing the user authentication and user-centric API routes, as well as bullet-proofing such with unit tests and actual real-world usage.
- Alberto: I was able to finish analyzing the theoretical material in the assigned lectures which allowed me to implement and test the cross-validation application.

How to address the issues in the next sprint?

- How to improve the process?
 - Alastair: Setup more frequent meetings (but currently suffering from some internet issues at my current venue).
 - Alberto: Improve communication between team mates throughout the sprint.
- How to improve the product?
 - Alastair: Less perfectionism, more prototyping speed first and actual implementation before polishing.
 - Alberto: Begin implementing functionality as soon as possible.

Sprint 4

Attendees: Alastair Paragas, Alberto Mizrahi

Start time: 07/09/2017, 12:00PM

End time: 07/09/2017, 1:00PM

What went wrong?

- Did we do a good job estimating our team's velocity?
 - Alberto Mizrahi: Yes we did. We were able to successfully complete all the user stories for this sprint.
 - Alastair Paragas: Yes.
- Did we do a good job estimating the points (time required) for each user story?
 - Alberto Mizrahi: Yes we did. Each user story took roughly the amount of time we approximated.
 - Alastair Paragas: Yes.
- Did each team member work as scheduled?
 - Alberto Mizrahi: Yes, every team member worked the hours required of him.
 - Alastair Paragas: Yes.

What went right?

- Alastair: We were able to finish all the user stories in time.
- Alberto: We were able to implement the API routes and the summarization algorithms and then successfully integrate them back to the application.

How to address the issues in the next sprint?

- How to improve the process?
 - Alastair: Set up more meetings.
 - Alberto: Continue communicating with the rest of the team members as much as possible.
- How to improve the product?
 - Alastair: Implements features first, then perfect them.
 - Alberto: Implements more features in a shorter amount of time.

Sprint 5

Attendees: Alastair Paragas, Alberto Mizrahi

Start time: 07/14/2017, 12:00PM

End time: 07/14/2017, 1:00PM

What went wrong?

- Did we do a good job estimating our team's velocity?
 - Alberto Mizrahi: Yes. We were successful in completing the given user stories during the Sprint.
 - Alastair Paragas: Yes.
- Did we do a good job estimating the points (time required) for each user story?
 - Alberto Mizrahi: Yes. Each user story took the amount of time it was expected to take.
 - Alastair Paragas: Yes.
- Did each team member work as scheduled?
 - Alberto Mizrahi: Yes. All the team members were responsible in finishing their work on time.
 - Alastair Paragas: Yes.

What went right?

- Alastair: I was able to implement the front-end and deploy the website.
- Alberto: We were able to successfully implement two more summarization algorithms and integrate them to the summarization API.

How to address the issues in the next sprint?

- How to improve the process?
 - Alastair: Do not let the tasks in the user story accumulate.
 - Alberto: Improve the team's communication and cooperation.
- How to improve the product?
 - Alastair: Implement more features.

- Alberto: Improve the efficiency and accuracy of the summarization algorithms.

Installation/Maintenance Document

The following document describes in detail how set up the Tidbit system in a local environment.

Retrieving the project from Github

The most up-to-date, stable release of Tidbit can be found in its GitHub repository:
<https://github.com/FIU-SCIS-Senior-Projects/Automated-Document-Summarization-1.0>

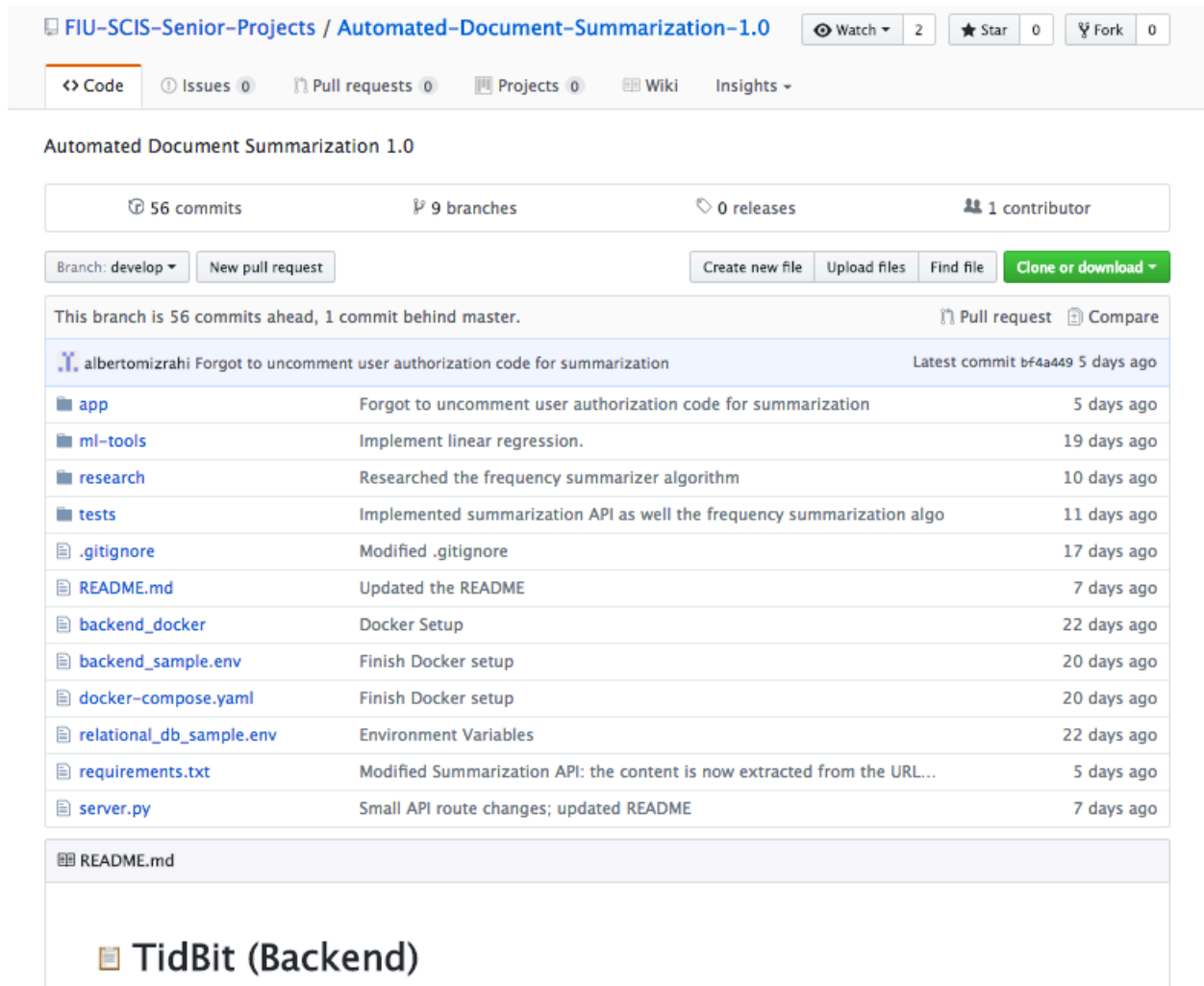


Figure D.11. A picture of the Tidbit GitHub repo.

After installing Git in your local system, follow the following steps to clone the GitHub repository to your local environment:

1. In your terminal application of choice, travel to the directory where you wish to download the project.
2. Input the following command into the terminal: `git clone https://github.com/FIU-SCIS-Senior-Projects/Automated-Document-Summarization-1.0.git`
3. Immediately thereafter, input the following command: `cd Automated-Document-Summarization-1.0/Code`

4. You are now in the project's root directory.

Setting up the backend locally

Our development workflow utilizes Docker and Docker Compose that allow us to compartmentalize our development environment so that it can be shared, set up and build in an easy and straightforward manner.

Before going it any further, you must install Docker and Docker Compose. Since the installation instructions are somewhat complex and vary between different computer system, operating systems and computer architectures, we provide links to the official guides on how to set them up:

- **Docker:** <https://docs.docker.com/engine/installation/>
- **Docker Compose:** <https://docs.docker.com/compose/install/>

Once they are installed, follow the following instruction to set up the local server:

1. Go into the project's Website directory and then go into the backend folder.
2. Rename backend_sample.env and relational_db_sample.env to backend.env and 3. relational_db.env
3. Open the backend.env file. You will see the following:

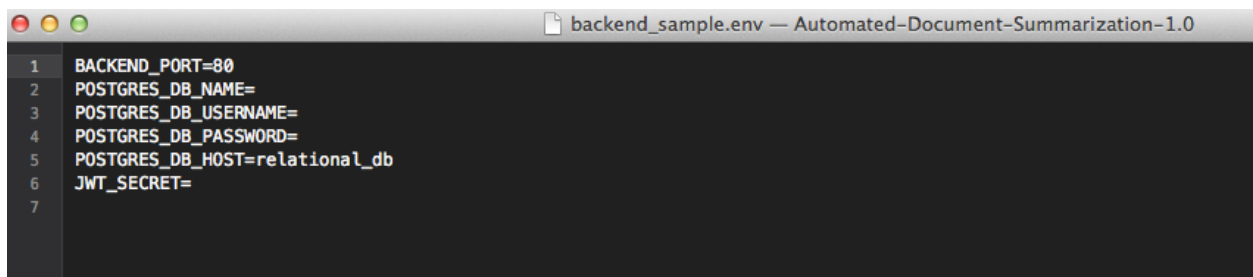


Figure D.12. Contents of the backend.env file.

4. Modify the contents of backend.env as you prefer. In particular:
 - BACKEND_PORT=80: you should not change this.
 - POSTGRES_DB_NAME: the name that you want to give to the system's database.
 - POSTGRES_DB_USERNAME: username for the account to access the database.
 - POSTGRES_DB_PASSWORD: password for the account to access the database.
 - POSTGRES_DB_HOST=relational_db: you should not change this.

- `JWT_SECRET`: the secret string that will be used to encrypt the database contents. It can be anything you want.
5. Open the `relational_db.env` file. You will see the following:

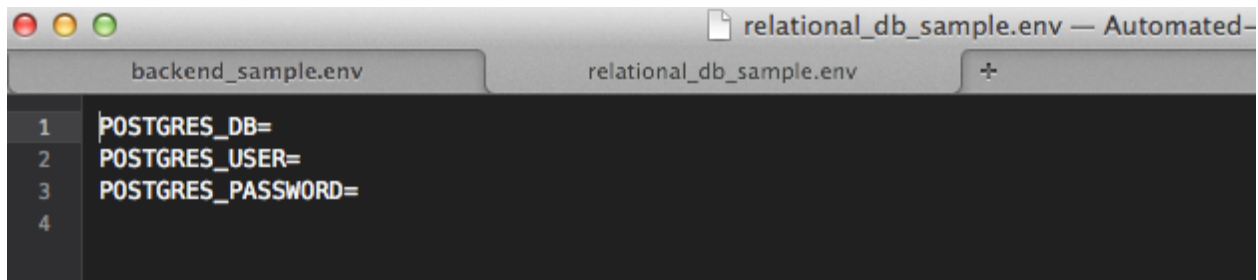


Figure D.13. Contents of the `relational_db.env` file.

6. Fill in for each entry with the same information as you did for the `backend.env` file. Notice that in the `relation_db.env` file, it says `POSTGRES_DB` instead of `POSTGRES_DB_NAME`. They are, however, the same.
7. Run the following command in the terminal (note: if you receive an error when running this command or the ones that follow, add the word “*sudo*” before each command): *docker-compose build*
8. After it finishes, run the command: *docker-compose up*. If you get any errors (they appear as yellow messages) while this command is executing, press Ctrl-C. Then go to Step 10 and then repeat from this step.
9. After a few seconds, the server should be now up and running. You should be able to access the application’s backend by going to the following URL: *http://localhost:8000/*
10. After finishing, do not forget to run the following command to kill the server and destroy the Docker container: *docker-compose down*

Setting up the frontend locally

Before proceeding please make sure that the latest versions of Node.js and NPM are installed by following the official guide: <https://docs.npmjs.com/getting-started/installing-node>. Once done, continue with the following steps:

1. Go the project’s root directory and then go to the frontend folder.
2. Run the following command: *npm install*. This will install all the dependencies needed to compile the front end

Hosting the website in your local environment

In order to access the website locally, please follow the following steps:

1. In a terminal, go to the backend folder and execute the command: *sudo docker-compose up*.
2. In a separate terminal, go to the frontend folder and execute the command: *npm run dev*.
3. In a separate terminal, go to the frontend folder and execute the command: *sudo python -m SimpleHTTPServer <port>*, where you replace *<port>* by the port number where the website will be hosted (note that port 80 can not be used as it is already busy with handling the backend).
4. You can now access the website locally by following the URL: <http://localhost:<port>>, again replacing *<port>* by the port number you specified in step 3.

Running the backend tests

In order to run the backend tests, proceed with the following steps:

1. In a terminal, go to the backend folder and execute the command: *sudo docker-compose up*.
2. In a different terminal, go to the backend folder and execute the command: *sudo docker exec -it backend_backend_1 python -m unittest tidbit/tests/<path to test file>* where *<path_to_test>* is replaced with the path to the test file, e.g. *'db/UserRepository_test.py'*.

Running the frontend tests

In order to run the frontend Selenium tests, proceed with the following steps:

1. In a terminal, go to the frontend folder and execute the command: *npm run test*

Shortcomings/Wishlist Document

After concluding with the first development iteration of this system, the team compiled the following list of features that should be considered for implementation during the next iteration:

- Allow users to recover their passwords.
- Allow for the existence of administrators which can disable or delete other user accounts.
- Increase the system's security by storing the passwords in the database in an encrypted manner.
- Improve the article extraction algorithm so as to improve the summarization accuracy.
- Implement more NLP algorithms, especially the more advanced ones.
- Modify the NLP algorithms so that they support other languages besides English.

REFERENCES

Erkan, Gunes, and Dragomir R. Radev. "LexRank: Graph-based Centrality as Salience in Text Summarization." *Journal of Artificial Intelligence Research* 22 (2004): n. pag. Web.

Haghighi, Aria, and Lucy Vanderwende. "Exploring Content Models for Multi-document Summarization." *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09* (2009): n. pag. Web.

Luhn, Hans Peter. "The Automatic Creation of Literature Abstracts." *IBM Journal of Research and Development* 2.2 (1958): 159-65. *The Automatic Creation of Literature Abstracts - IEEE Xplore Document*. Web. 14 July 2017.

Mihalcea, Rada, and Paul Tarau. "TextRank: Bringing Order into Texts." *Proceedings of EMNLP* (2004): 404-11. Web.

Steinberger, Josef, and Karel Jezek. "Update Summarization Based on Latent Semantic Analysis." *Text, Speech and Dialogue Lecture Notes in Computer Science* (2009): 77-84. Web.

Vanderwende, Lucy, Hisami Suzuki, Chris Brockett, and Ani Nenkova. "Beyond SumBasic: Task-focused Summarization with Sentence Simplification and Lexical Expansion." *Information Processing & Management* 43.6 (2007): 1606-618. Web.