

# Progress Report 2

## Web Summarizer and Shortener

COSC 4P02  
March 25th, 2024  
Naser Ezzati-Jivan

Tanvir Hasan - <i>Product Owner</i>	(6599328)	<a href="mailto:th18ai@brocku.ca">th18ai@brocku.ca</a>
Muhammed Bilal - <i>Scrum Master</i>	(6695738)	<a href="mailto:bb18hb@brocku.ca">bb18hb@brocku.ca</a>
Hamza Sidat - <i>Developer</i>	(6599591)	<a href="mailto:hs18so@brocku.ca">hs18so@brocku.ca</a>
Marium Nur - <i>Developer</i>	(7327182)	<a href="mailto:mn21xu@brocku.ca">mn21xu@brocku.ca</a>
Anjali Sabu - <i>Developer</i>	(7337033)	<a href="mailto:as21qj@brocku.ca">as21qj@brocku.ca</a>
Abdul-Maalik Siddiqui - <i>Developer</i>	(6785828)	<a href="mailto:as19fa@brocku.ca">as19fa@brocku.ca</a>
Amani Anderson - <i>Developer</i>	(6617344)	<a href="mailto:aa18ex@brocku.ca">aa18ex@brocku.ca</a>

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>1. Introduction</b>	<b>3</b>
<b>2. Design</b>	<b>3</b>
2.1. Summarizer	3
2.2. URL Shortener	4
2.3. Update to Create Account and Forgot Password	4
2.4. Addition of error messages for authentication pages	5
2.5. User Dashboard	6
2.5.1. History	6
2.5.2. Templates	6
2.5.3. Settings	7
2.6. Dark Mode	7
<b>3. Implementation</b>	<b>10</b>
3.1. Text Extraction	10
3.1.1. Web Text Extraction	10
3.2. Summarizer (addition of YouTube video summarization & custom features like bullet points, tones, etc)	10
3.2.1. Summarizer Tests	12
3.3. URL Shortener	13
3.3.1. URL Shortener Tests	13
3.4. PostgreSQL Database	13
3.5. User History	15
3.5.1. History Management	15
3.5.2. User History Management Tests	15
3.6. YouTube Text Extraction for full and part of the video	15
3.6.1. Fetch the caption	15
3.6.2. Test cases	16
3.7. Account Creation	16
3.8. Login with Google and Facebook	17
3.8.1. Google	17
3.8.2. Facebook	17
3.9. User Management	18
3.9.1. User Management Testing	18
<b>4. Release Planning</b>	<b>19</b>
4.1. Sprint 3 (23 Feb - 8 Mar)	19
4.2. Sprint 4 (8 Mar - 22 Mar)	20
<b>5. Problems</b>	<b>22</b>
5.1. User login issue when page refreshes	22

5.2. URL will not shorten without HTTP	22
<b>6. Future Steps</b>	<b>22</b>
6.1. Frontend	22
6.2. Backend	23
<b>7. Meeting Notes</b>	<b>23</b>

# 1. Introduction

Since our last progress report, Shortify has made significant strides. We now have a nearly complete product, packed with features for summarization, link shortening, and premium offerings. We have successfully integrated the back-end with a user-friendly front-end interface of the web application, resulting in the addition of numerous new features. This report will detail our achievements, including the addition of new features, quality improvements, and ease-of-use functionality. It will also discuss our testing strategies for each functionality.

## 2. Design

### 2.1. Summarizer

The screenshot displays the Shortify Summarizer interface. At the top, there are tabs for 'Text', 'Website URL', and 'Youtube URL'. Below these, there are 'Modes' (Standard and Paragraph) and a 'Summary Length' slider (Short to Long). A 'Save settings' button is in the top right. The main text area contains a paragraph about the Postgres database migration. A red box at the bottom left indicates 'Get Premium for unlimited words. 164/125 Words'. A 'Summarize' button is at the bottom right. The output area on the right is empty, with the placeholder text 'Get summary here...'.

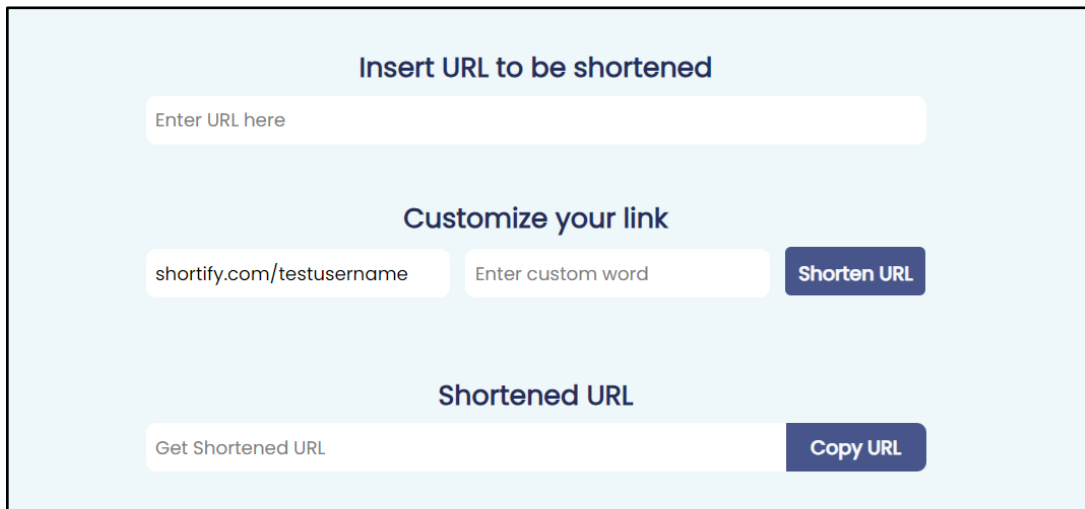
The summarizer component has been updated with the addition of modes that only premium users can access. The modes panel would allow users to select the tone, formatting (bullet or numbered points) and length of the summarized output. The summarizer also has a word count feature which updates as the user types the input. It also shows the red box (displayed above) indicating users can get a premium account if they exceed 125 words and have not yet created an account, while also disabling the “Summarize” button to prevent the user from further use.

This screenshot shows an updated version of the Shortify Summarizer interface. It includes additional settings: 'Timestamp' (a dropdown menu), 'Start Time' (HH and MM input fields), and 'End Time' (HH and MM input fields). The 'Summarize' button is now disabled. The text input area contains the placeholder text 'Enter or paste your Youtube URL and click "Summarize."'. The output area on the right is empty, with the placeholder text 'Get summary here...'.

For the summarization of a YouTube video, users can choose to summarize the full video or part of the video by specifying the video timestamps.

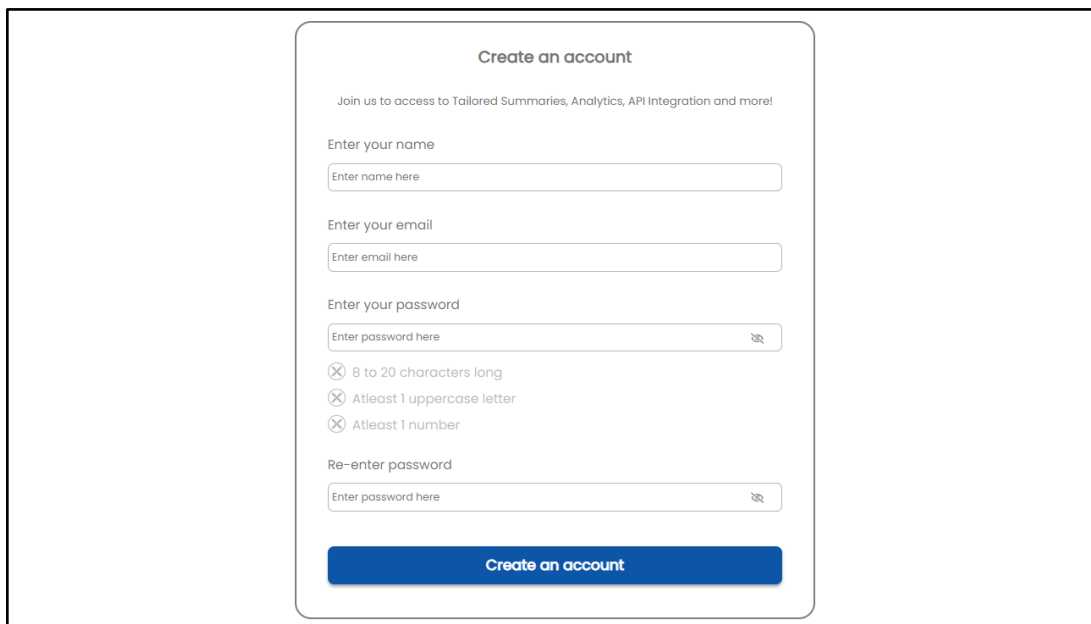
## 2.2. URL Shortener

This page is for premium users to shorten a URL. It allows them to shorten a URL and include a custom word in the shortened link.



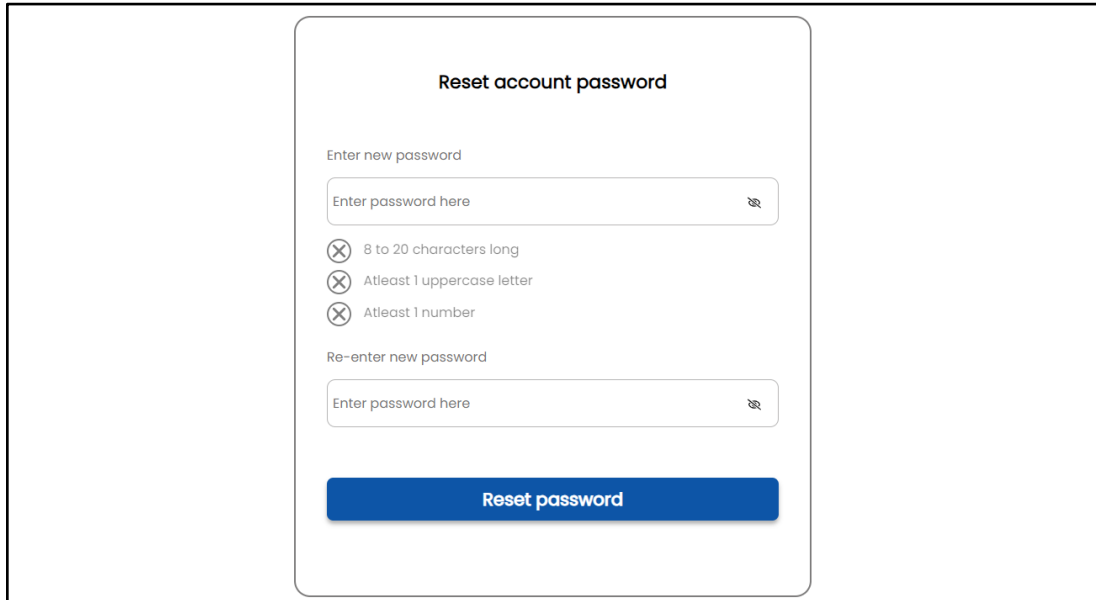
The screenshot shows a light blue background with three main sections. The first section, titled "Insert URL to be shortened", contains a white input field with the placeholder text "Enter URL here". The second section, titled "Customize your link", features a white input field containing "shortify.com/testusername", another white input field with the placeholder "Enter custom word", and a dark blue button labeled "Shorten URL". The third section, titled "Shortened URL", has a white input field with the placeholder "Get Shortened URL" and a dark blue button labeled "Copy URL".

## 2.3. Update to Create Account and Forgot Password



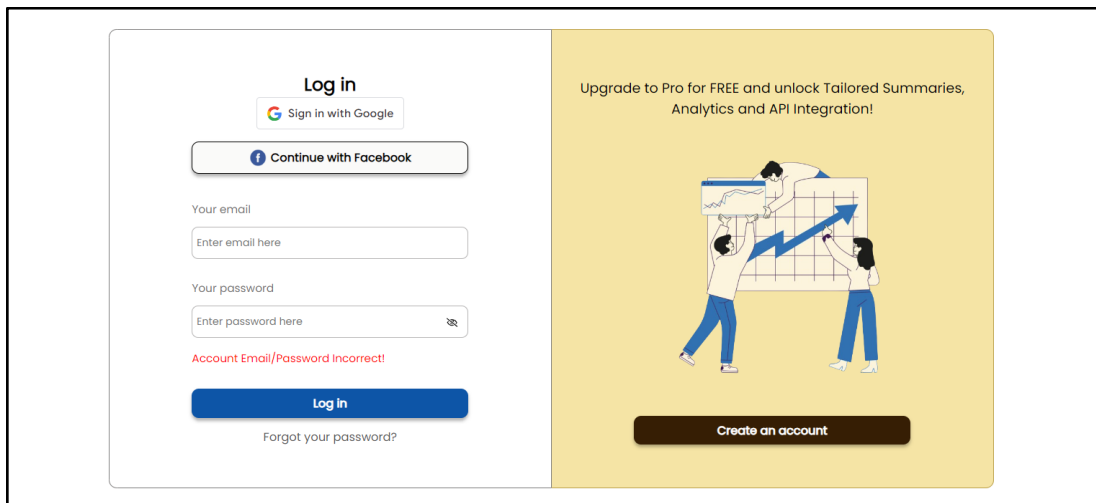
The screenshot displays a "Create an account" form on a light gray background. The form is enclosed in a white rounded rectangle with a thin gray border. At the top, it says "Create an account" in bold, followed by a smaller line of text: "Join us to access to Tailored Summaries, Analytics, API Integration and more!". Below this are four input fields: "Enter your name" (placeholder: "Enter name here"), "Enter your email" (placeholder: "Enter email here"), "Enter your password" (placeholder: "Enter password here" with a toggle icon), and "Re-enter password" (placeholder: "Enter password here" with a toggle icon). Between the password fields are three checkboxes, each with a red 'X' icon and the text: "8 to 20 characters long", "Atleast 1 uppercase letter", and "Atleast 1 number". At the bottom of the form is a large blue button labeled "Create an account".

Both the create account and forgot password pages have been updated with a password requirements checklist which automatically checkmarks as the user types the password with the specified requirements. We have also added a hidden password (eye) icon as an extra safety measure.



The image shows a 'Reset account password' form. At the top, it says 'Reset account password'. Below that is a label 'Enter new password' followed by a text input field with the placeholder 'Enter password here' and a small eye icon on the right. Under the input field is a checklist with three items, each with a circled 'X' icon: '8 to 20 characters long', 'Atleast 1 uppercase letter', and 'Atleast 1 number'. Below the checklist is a label 'Re-enter new password' followed by another text input field with the placeholder 'Enter password here' and an eye icon. At the bottom of the form is a blue button labeled 'Reset password'.

## 2.4. Addition of error messages for authentication pages



The image shows a 'Log in' form. At the top, it says 'Log in'. Below that are two social login buttons: 'Sign in with Google' and 'Continue with Facebook'. Below these are two text input fields: 'Your email' with placeholder 'Enter email here' and 'Your password' with placeholder 'Enter password here' and an eye icon. Below the password field is a red error message: 'Account Email/Password incorrect!'. Below the error message is a blue button labeled 'Log in'. At the bottom of the form is a link: 'Forgot your password?'. To the right of the form is a yellow promotional banner. The banner has the text 'Upgrade to Pro for FREE and unlock Tailored Summaries, Analytics and API Integration!'. Below the text is an illustration of three people working on a large grid with a blue arrow pointing upwards. At the bottom of the banner is a dark blue button labeled 'Create an account'.

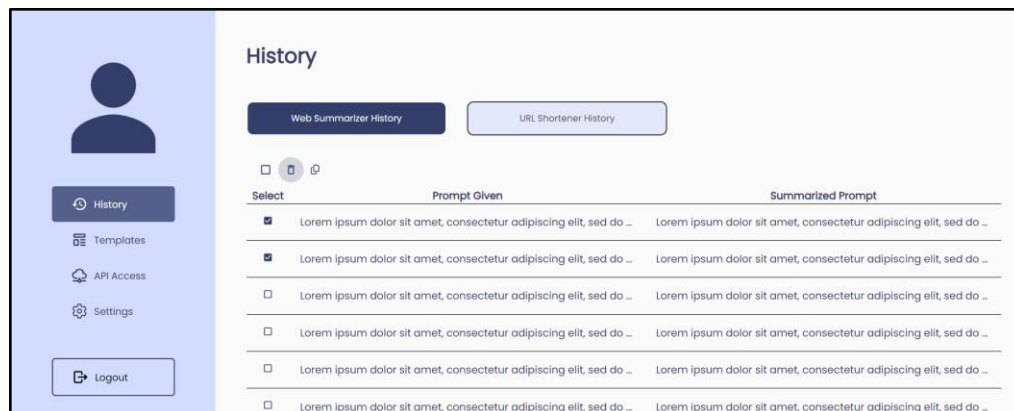
We have error messages for the create account and log-in pages, such that users are informed when they enter an invalid email or password, and when a user account does not exist. Backend functionalities have also been added such that error messages appear when required.

## 2.5. User Dashboard

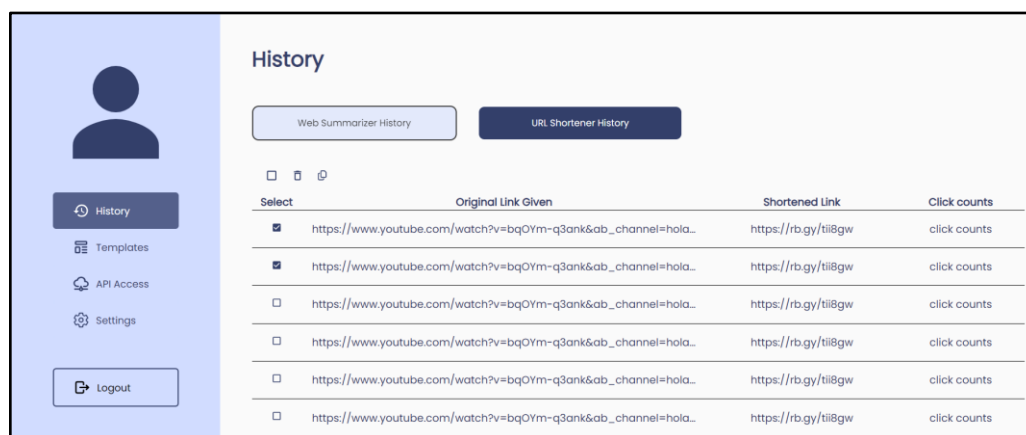
The User Dashboard has a sidebar with four menu options, including History, Templates, API Access and Settings and a Logout button.

### 2.5.1.

### 2.5.2. History



The History page in the User Dashboard will display their previous work using the Summarizer and the URL Shortener, allowing users to copy the summarized text or delete the prompt/record altogether in the case of the Summarizer and copy the shortened URL or delete the prompt/record altogether in the case of the URL Shortener. Additionally, the URL Shortener history displays click counts of the new URL. The History page needs to be integrated with the backend, which is a task for the last sprint.



### 2.5.3. Templates

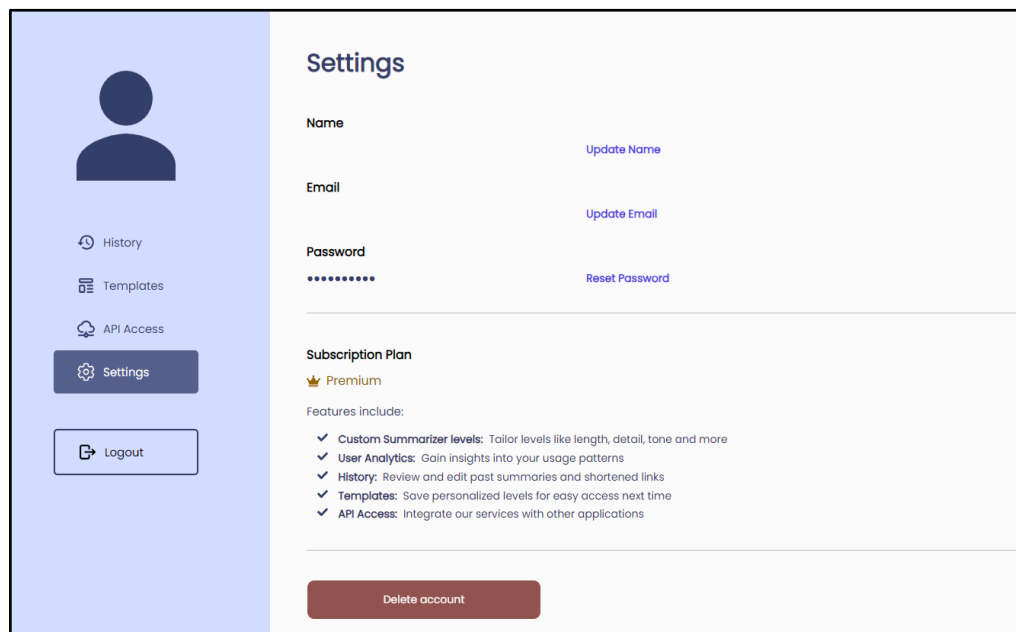
The Templates page displays up to three custom templates that the user can save to be used (loaded) if they require the same settings the next time they use the

Summarizer. Each template has Save and Reset buttons for the user to change their saved template if needed. Also required backend integration.



#### 2.5.4. Settings

The Settings page allows users to change their name and email associated with the account and reset their password along with giving them the option to delete their account which would then remove the user from our database.



## 2.6. Dark Mode

The dark mode has been implemented for the Summarizer, URL Shortener, Feedback, and Homepage pages. The rest of the sprint will focus on implementing the same for all pages of the website.





shortify

SUMMARIZER

URL SHORTENER



# Your shortcut to simplicity.

Say goodbye to lengthy URLs and information overload — discover the power of simplicity with our intuitive platform.

Try it now!



## Summarizer

Experience instant clarity with our Summarizer tool!

Sign up now to access pro Summarizer features like customizable summarization levels, API Access and more!



## URL Shortener

Amplify your impact with our URL Shortener tool!

Perfect for social media posts, our tool ensures your content gets noticed without cluttering your message.



This website streamlines text summarization with precision.



Brilliantly effective! This website nails the art of summarization with finesse.



Time-saving gem! Say goodbye to long hours of reading



[Help](#) [Feedback](#)

### Summarizer

Text

Website URL

Youtube URL

Modes:

Standard

Paragraph

Summary Length:

Short

Long

Save settings

Enter or paste your text and click "Summarize."

Get summary here...

0 Words

Summarize

### Insert URL to be shortened

Enter URL here

### Customize your link

shortify.com/testusername

Enter custom word

Shorten URL

### Shortened URL

Get Shortened URL

Copy URL

### Rate your experience

We highly value your feedback! Kindly take a moment to rate your experience and provide us with your valuable feedback.

Tell us about your experience

Send

## 3. Implementation

### 3.1. Text Extraction

#### 3.1.1. Web Text extraction

No new features were added in this part of the project.  
See Progress Report 1

### 3.2. Summarizer (addition of YouTube video summarization & custom features like bullet points, tones, etc)

The summarizer has finally been integrated with the front end, and it includes many new features. The summarizer's main text summary feature will take in user-provided text, and give them a summary. The summary will be given based on the user's selection of tone, style and length as implemented at the top of the tool's window.

The screenshot displays the Summarizer tool interface. At the top, there are three tabs: 'Text', 'Website URL', and 'Youtube URL'. Below the tabs, there are two dropdown menus for 'Modes' (Standard and Bullet Points) and a 'Summary Length' slider (Short to Long). A 'Save settings as a template' button is located on the right. The main text area contains a paragraph about artificial intelligence (AI) and deep learning. To the right of the text area, there is a list of bullet points summarizing the text. At the bottom left, it says '99 Words'. At the bottom center, there is a 'Summarize' button. At the bottom right, there are icons for sharing and a comment bubble.

The summarizer tool now uses OpenAI's gpt-4-0125-preview model, which allows for a much larger context window of 128,000 tokens, and a higher-quality summary. The web page summarizer has now been implemented with the front end and works like before, only now with support for websites with much more content, without the fear of running out of tokens like in the gpt-3.5-turbo-0125 model.

Text
Website URL
Youtube URL

Modes:
Formal
Numbered List

Summary Length:
Short
Long

Save settings as a template

https://en.wikipedia.org/wiki/Main\_Page

1 Word

Summarize

Shorten your URL

1. John Littlejohn was an American tradesman and Methodist preacher who migrated to the Thirteen Colonies from Great Britain. He preached as a circuit rider during the Revolutionary War and faced persecution from colonial authorities despite his revolutionary sympathies before settling in Leesburg, Virginia, where he served as a local preacher and tradesman. Littlejohn later moved to Kentucky, working as a land agent and passing away in 1836 at his home in Logan County.

2. The Francis Scott Key Bridge in Baltimore collapsed after being hit by a container ship, resulting in a tragic event. Bassirou Diomaye Faye was elected President of Senegal, and a mass shooting and explosions in Krasnogorsk, Russia, led to the loss of 143 lives. Prabowo Subianto won the presidential election in Indonesia, and ongoing events include the Haitian crisis, Israel-Hamas war, Myanmar civil war, Red Sea crisis, and the Russian invasion of Ukraine.

3. On March 30, various historical events took place, such as the merging of East Florida and West Florida to create the Florida Territory in 1822, and the signing of the Treaty of Fes by Sultan Abd al-Hafid in 1912, making Morocco a French protectorate. Annie Hall had its first screening in 1977.

The tool now also works with an improved version of YouTube summarization, by making use of a new approach to retrieving the content of a YouTube video. The summarizer will now, like before, retrieve the captions of the video for summarization, but now, unlike before, where if the captions were disabled or not working the summary would fail, we will download the audio, and transcribe the video's audio, to then be summarized by the AI.

Text
Website URL
Youtube URL

Modes:
Formal
Paragraph

Summary Length:
Short
Long

Full Video

Save settings as a template

https://www.youtube.com/watch?v=NHopJHSIVo4

1 Word

Summarize

Shorten your URL

In a thought-provoking video, the speaker addresses the phenomenon where sharing your goals with others can actually hinder your likelihood of achieving them. Psychologists have found that when you vocalize your goals and receive acknowledgment, your mind may mistake the act of talking for the actual work needed, leading to decreased motivation. A study involving 163 participants demonstrated that those who kept their goals private worked longer towards them compared to those who announced their goals. The speaker suggests resisting the urge to announce goals, delaying the gratification of social acknowledgment, and being cautious about how goals are shared to maintain motivation. The audience is left with the insightful message to carefully consider how they communicate their goals to avoid undermining their achievement.

Reference:  
TEDx Talks. (2010, September 06). Derek Sivers: Keep your goals to yourself [Video]. YouTube. https://www.youtube.com/watch?v=NHopJHSIVo4

Finally, as mentioned before, all tones (see photo below for all options) formats for the summary (bullet points, paragraph, numbered list), and length of the summary (short to long) are all functional. The summarizer now also has the ability to summarize a YouTube video in a time range and also now includes a citation feature which can be toggled and a choice of citation format (APA, MLA, Chicago, etc..) is available, although yet to be implemented on the front end as UI, but is functional. The citation tool can cite both the YouTube video URL and the given Web Page URL in any citation Format. As seen in the other photos, the way one likes their summary can also be saved as a setting template, which they can save many different versions of, so that the user does not need to choose their selections each time.

The screenshot shows the Summarizer application interface. At the top, there are three tabs: 'Text', 'Website URL', and 'Youtube URL'. The 'Youtube URL' tab is selected. Below the tabs, there are several controls: 'Modes' with a dropdown menu showing 'Formal' (selected), 'Paragraph' (selected), and 'Summary Length' with a slider between 'Short' and 'Long'. There is also a 'Timestamp' dropdown and 'Start Time' and 'End Time' fields with HH:MM inputs. The main input area contains a YouTube URL: 'https://www.youtube.com/watch?v=NHopJHSIVo4'. A dropdown menu is open over the URL, showing options: 'Standard', 'Formal', 'Causal', 'Sarcastic', 'Aggressive', and 'Sympathetic'. The 'Formal' option is selected. The output area on the right displays a paragraph summarizing the video content. Below the summary, there is a 'Reference' section with the following text: 'TEDx Talks. (2010, September 06). Derek Sivers: Keep your goals to yourself [Video]. YouTube. https://www.youtube.com/watch?v=NHopJHSIVo4'. At the bottom, there are two buttons: 'Summarize' and 'Shorten your URL'. The bottom left corner shows '1 Word' and the bottom right corner has social media icons.

### 3.2.1. Summarizer Tests

The summarizer test makes use of unittest to check to make sure each function in the back end is working as intended. The tests check to ensure each tone is functional, each style is functional, the length slider is functional if empty text gives an error, and if each feature is connected as intended with their respective functions. The text, website URL, and YouTube URL tabs are all working as they should.

Tests for retrieving web page text content and YouTube video transcripts are discussed in their respective sections. This test only tests if the summary itself and all its features work as we intend. The test works by sending a json file as it would receive from the front end, and testing to ensure each feature works and either returns a json with the summary or a json with an error where appropriate.

```

(venv) amws@Ams-MacBook-Air server % python -m unittest testServer.py
tone: Standard
style: Paragraph
length: short
option: Full Video
cite: APA

.tone: Standard
style: Paragraph
length: short
option: Full Video
cite: APA

User pasted text: Sample text for summarization
.tone: Formal
style: Bullet Points
length: medium
option: Full Video
cite: APA

User given URL https://en.wikipedia.org/wiki/Inner_Harbor
.tone: Causal
style: Numbered List
length: long
option: Full Video
cite: APA

User given YouTube URL https://www.youtube.com/watch?v=NHopJHSLVo4
[youtube] Extracting URL: NHopJHSLVo4
[youtube] NHopJHSLVo4: Downloading webpage
[youtube] NHopJHSLVo4: Downloading ios player API JSON
[youtube] NHopJHSLVo4: Downloading android player API JSON
WARNING: [youtube] Skipping player responses from android clients (got player responses for video "aQvGIIdgFDM" instead of "NHopJHSLVo4")
[youtube] NHopJHSLVo4: Downloading m3u8 information
.

Ran 4 tests in 20.430s

OK

```

### 3.3. URL Shortener

The URL Shortener's backend implementation remains largely unchanged. One change made to the implementation since the previous progress report is the handling of links that do not contain https (or http) before them. Understandably, some users may just type in [www.youtube.com](http://www.youtube.com) instead of <https://www.youtube.com> when shortening their link. This type of input should be handled, and previously would return an error when the Python redirect function attempts to redirect to the original link; the link without https:// as a prefix. This error was fixed by checking for the http or https pattern using a regex, and appending http:// to the link if it is given without such a prefix.

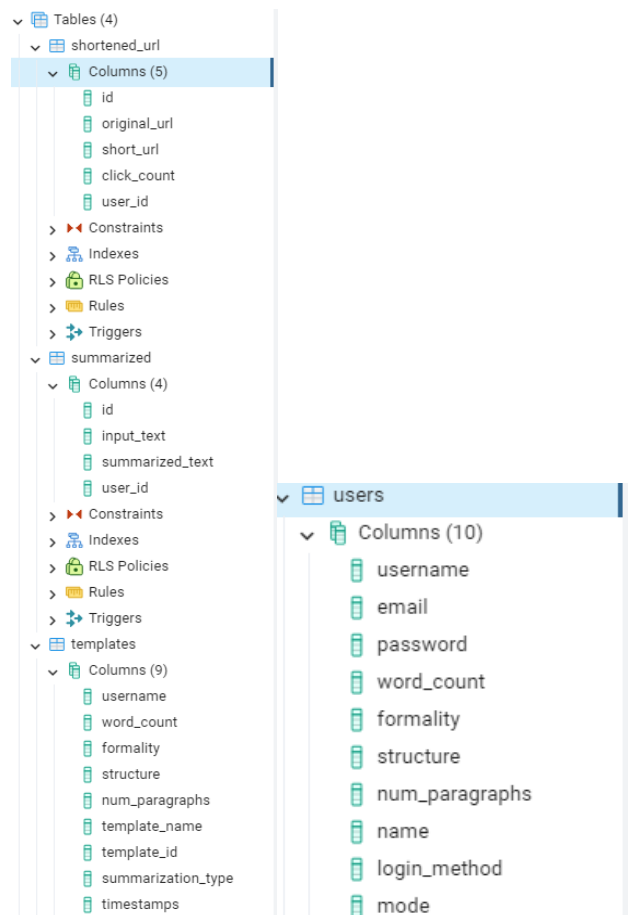
#### 3.3.1. URL Shortener Tests

The URL Shortener remained unchanged from the previous progress report, as remained mostly unchanged throughout the sprints since that Progress Report 1.

1. Test that the table exists in the database.
2. Test that a link is successfully shortened.
3. Test that a link is resolved (returns the original)
4. Test that the click count of a link is initially set to 0.
5. Test that the click count is updated when it is clicked.
6. Test that URLs are unique given that the same original URL is inputted.
7. Test that a custom URL is made.

### 3.4. PostgreSQL Database

The Postgre Database has evolved since the first Progress Report. Firstly, the database was migrated from a locally hosted Postgre database on Amani's local machine to a cloud-based database. What this means is, that the functionality that relies on the database can be accessed without relying on a local machine's internet access or having the database installed and configured on each user's machine. The cloud solution chosen is <https://www.elephantsql.com/> - which offers a free cloud database with 20MB. We notice that this is not a substantial amount of storage, so we will explore upgrading as needed. Additionally, the database includes two new tables, as well as changes made to the other two tables. There are now 4 tables in the database; shortened\_url - containing all URL shortener logic and information, users - containing all user information, summarized - containing the summarization information, and templates - containing the custom summarization levels that a user may save. Below is a screenshot of the database from pgAdmin.



It is important to note that these columns are likely to change, and are not final, as it is a work in progress and there are some requirements that we may add, or drop accordingly. However, the general idea of the database remains the same.

### 3.5. User History

### 3.5.1. History Management

User history is managed on our PostgreSQL Database, where a table is created to manage users' history. This table contains four columns, `history_id`, `input_text`, `summarized_text`, and `username`. We are using the `psycopg2` library to manage queries and to communicate with the database. The user history management component includes three functions, which are, insert history, retrieve history, and delete history. In insert history, there is a query to check before storing the data, if the user has over five entries associated with their username. If the table contains more than five entries, then the oldest entry is removed and the new entry is stored. History is retrieved given the username, where input text and summarized text associated with the given username will be retrieved. Lastly, to delete history, there is an option to either delete history by username, where all entries associated with a given username will be deleted, or by history ID, where only specific entries for which ID has been given, will be deleted.

### 3.5.2. User History Management Tests

Unit tests for user history management backend code were performed using `unittest` library in Python, which included, inserting, retrieving, and deleting entries with and without errors, which were all passed.

## 3.6. Youtube Text Extraction for full and part of the video

### 3.6.1. Fetch the caption

To get the caption from YouTube, we have used libraries like `yt_dlp`: for downloading video and extracting audio, `boto3`: to facilitate interaction with AWS S3 for storing audio files and AWS Transcribe for converting audio to text, `youtube_transcript_api`: which provides an alternative method to fetch YouTube captions when available, reducing processing time directly, `ffmpeg`: to trim audio files to the specified segments, ensuring that only relevant portions are processed for transcription.

The workflow includes several steps:

1. Extract the YouTube video ID from the provided URL.
2. Validating and converting specified time segments from HH: MM format to seconds.
3. Downloading the entire audio track or specified segments.
4. Uploading audio files to AWS S3.
5. Submitting audio files for transcription through AWS Transcribe.
6. Fetching and compiling the transcription results.

Upon giving the YouTube URL and the option between full or part of the video, the system will then try to see if the video has captions or not. If the video has captions,



it will give the captions immediately. If not, then it will download the video and convert it to an audio file. The audio file will then be uploaded to AWS S3. Using AWS Transcribe, it will then use the uploaded audio to transcribe it and return it to the user. The trimming function is used for the audio to trim to the specific part to transcribe.

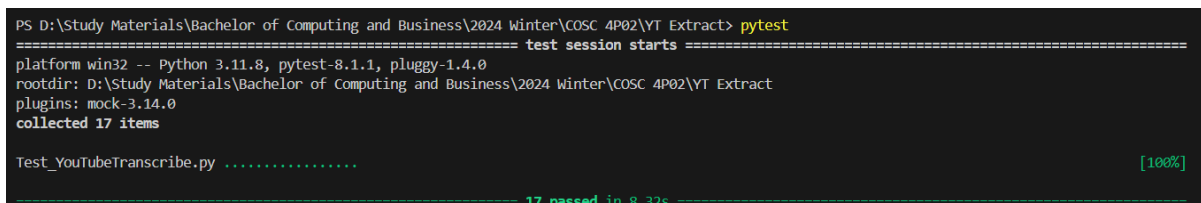
### 3.6.2. Test cases

The testing suite utilizes pytest.

Key Components Tested:

1. Time Conversion (convert\_time\_to\_seconds): Validates the correct conversion of time strings (HH:MM) to seconds.
2. Time Validation (validate\_time\_within\_duration): Ensures start and end times are correctly validated against the video's duration.
3. Audio Downloading (download\_audio): Verifies that audio files are correctly downloaded from YouTube using the yt\_dlp library. Mocking is used to simulate the download process and verify the function's response.
4. Audio Transcription (transcribe\_audio): Tests the transcription process facilitated by AWS Transcribe, checking that the function handles the transcription lifecycle correctly and returns the expected text.
5. File Uploading (upload\_file\_to\_s3): Confirms that audio files are successfully uploaded to AWS S3, and handles upload failures appropriately.

The system passed all the test scenarios successfully. Below is a screenshot of that.



```
PS D:\Study Materials\Bachelor of Computing and Business\2024 Winter\COSC 4P02\YT Extract> pytest
===== test session starts =====
platform win32 -- Python 3.11.8, pytest-8.1.1, pluggy-1.4.0
rootdir: D:\Study Materials\Bachelor of Computing and Business\2024 Winter\COSC 4P02\YT Extract
plugins: mock-3.14.0
collected 17 items

Test_YouTubeTranscribe.py ..... [100%]

===== 17 passed in 8.32s =====
```

## 3.7. Account Creation

Account creation is a core part of this project. Our pro features are essentially simulated by having a user create an account. If you create an account, you are a pro-user and you have access to all of the features. There are three ways a user can create an account; manually, using an email address, with their password and a selected nickname, or using Facebook/Google authentication. This section will focus on manual creation. When a user attempts to create an account, they are prompted to enter an email address, a name and a password. There are several cases to consider, all of which are handled. For instance, we integrated a free email-verification API

(<https://api.hunter.io/v2/email-verifier>) that will check to see if an email is valid when a user attempts to register with it. This API allows for 50 free calls a month, however, so it is disabled in our testing version as we did not want to waste it while testing. If a user enters an email that's already registered, an error will appear, if a user enters an invalid password, given that it doesn't meet the requirements (8-20 chars, 1 uppercase, 1 number), an error will appear. If the password and re-entered password don't match, an error will appear, etc. When a user enters the correct information, a fetch is made to our Flask backend running on a localhost port 5001. Here, the information is processed and fed to the database, with JSON return statements that allow for error handling and routing. Currently, the user does not enter a 'username', but rather, a 'name', which is not necessarily unique to the user. The user's only unique identifier (that is known to them) is their email. In the backend, each email that is entered is given a unique 6-character hash as a username, which allows for more fluid backend processing. This serves as the backend's unique identifier for most cases. When a user account is created, three rows are created in the templates table in the database, corresponding to that user account. This is because each user is given three custom templates - corresponding to three sets of permutations of summarization levels that they can save into a given template. When an account is deleted, these rows are naturally deleted as well.

## **3.8. Login with Google and Facebook**

### **3.8.1. Google**

Logging in with Google makes use of the Google Cloud Console's OAuth API. The API was integrated into our front end through its provided functions to create a button and allow user sign-ins. In the front end, we make use of the `renderButton` function that the API provides, which upon clicking, allows a user to select which account they want to login with. This sends a callback response, which contains an encoded JWT, from which the email and name is extracted. This information is then recorded in our database, and the user is then logged in by caching their email and name in local storage until they log out.

### **3.8.2. Facebook**

Facebook authentication has not been implemented yet, as is currently in progress.

## **3.9. User Management**

Associated with account creation, we allow users to make several changes to their accounts. They can delete their account, change their name, change their email

address, change their password, or reset their password if they forgot it. All of these cases have been implemented and tested. Each capability is represented in the front end, and when the user interacts with a given button, fetches are made to the Flask server, which is then routed to one of several defined functions that handle the logic behind the desired functionality. All user management functionality is handled by one Python file, which runs on a dedicated Flask server on a given port (currently 5001). These functions that are routed by Flask, all rely on the Authentication class within the Python file (authentication.py). There, SQL queries are defined and executed, which is where the database interaction takes place. Naturally, once a result is returned, this is fed back in the opposite direction and handled by the front end.

### 3.9.1. User Management Testing

Testing is imperative with a complex system such as this. Like the URL Shortening functionality, unit tests using Pytest were implemented to test the functionality of the user management class. The user management functionality is not complete, however, so the implemented tests are not final. Below are some of the tests that have been implemented so far:

1. Test that a user is registered
2. Test the 'checkIfAlreadyRegistered' method
3. Test the 'findEmail' method
4. Test that a user can 'login'
5. Test the 'isPasswordValid' function
6. Test account deletion
7. Test password-changing functionality
8. Test password reset functionality
9. Test change email functionality
10. Test getName functionality
11. Test name-changing functionality
12. Test get username functionality
13. Test isPasswordCorrect functionality
14. Test template row creation
15. Test template addition
16. Test template clearance
17. Test template deletion

```
(base) C:\Users\amani\OneDrive\Documents\BrockU\Code Stuff\COSC-4P02-Web-Summarizer\WebSummarizerShortner\server>pytest
testAuthentication.py
===== test session starts =====
platform win32 -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\amani\OneDrive\Documents\BrockU\Code Stuff\COSC-4P02-Web-Summarizer\WebSummarizerShortner\server
plugins: anyio-3.5.0
collected 16 items

testAuthentication.py ..... [100%]

===== 16 passed in 2.15s =====
```

More tests to come as the authentication functionality grows.

## **4. Release Planning**

### **4.1. Sprint 3 (23 Feb - 8 Mar)**

During Sprint 3, significant progress was made for various assigned tasks. We began with a productive meeting with the TA to review our progress and completed the first progress report, during which it was suggested to include upcoming features for future sprints in our report. Our focus on responsiveness paid off as most pages and components are now fully responsive across different screen sizes. The summarizer component saw notable enhancements, including custom features for premium users, additional input options such as text, website, and YouTube URLs, as well as hover effects and tooltips. Backend improvements were made to enable summarization with bullet points and other formatting options. Furthermore, users can now add timestamps for video summarization. Additionally, a URL shortener was successfully implemented and the branch was then merged with the User Interface branch. ElephantSQL is being used for the database, with numerous tables added to store history, summarizer inputs and outputs, URL Shortener inputs and output, and much more. The backend functionalities for history access, editing, and deleting are also working as of now alongside the addition of account management features like changing email and password, with enforced password requirements. Backend logic was refined to generate a hash-based username from user emails. Frontend error messages were implemented in the user account authentication process for better user guidance, supported by corresponding backend functionalities.

Key :	Summary :	Issue type :	Epic :	Status :	Assignee
SCRUM-161	Add inputs for website URL and YouTube links in the summarizer component	Task		DONE	MB
SCRUM-148	Ensure Hero component responsiveness for all screen sizes	Task		DONE	AS
SCRUM-154	Ensure URL Shortener component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-149	Ensure LandingPage component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-147	Ensure Feedback component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-10	As a user, I want to copy or export the summarized text so that I can save the text into my wo...	Story	STANDARD SUMMARIZATION	DONE	
SCRUM-157	Ensure Footer component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-156	Ensure Header component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-162	Add call-to-action buttons in the Hero section	Task		DONE	AS
SCRUM-163	Add hosting for the Postgres database	Task		DONE	AA
SCRUM-64	Add export functionality for the summarized text	Task		DONE	MS
SCRUM-165	Test youtube summarization	Task		DONE	MS
SCRUM-24	As a pro user, I want to be able to create an account with my email so that I can use the vario...	Story	USER DASHBOARD	DONE	AA
SCRUM-14	As a pro user, I want the summary to be formatted using bullet points, paragraphs and altern...	Story	CUSTOM SUMMARIZATION	DONE	MS
SCRUM-21	As a pro user, I want to choose the tone of language used in the output text between formal ...	Story	CUSTOM SUMMARIZATION	DONE	MS
SCRUM-69	Create a password reset functionality	Task		DONE	AA
SCRUM-72	Create a functionality to allow users to input specifications for text formatting	Task		DONE	MS
SCRUM-70	Add a functionality to change username/email	Task		DONE	AA
SCRUM-75	Add a log in and log out functionality	Task		DONE	AA
SCRUM-79	Create a functionality to choose between informal or formal summarized text output	Task		DONE	MS
SCRUM-82	Create a log in with email option	Task		DONE	AA
SCRUM-85	Add a feature to delete account	Task		DONE	AA
SCRUM-97	Allow users to input specific timestamps of a video input for summarization	Task		DONE	TH
SCRUM-170	re-adjust the Summarizer page buttons	Task		DONE	MB
SCRUM-172	Making components pretty, ensuring fonts, buttons and colors are uniform across components	Task		DONE	AS
SCRUM-173	Fix Minor UI issues with Summarizer - positioning, hover features, and borders	Task		DONE	MB
SCRUM-174	Make "Try it now" button clickable for the entire button/div	Task		DONE	AS
SCRUM-175	Add and change state visibility of textareas depending on menu click	Task		DONE	MB
SCRUM-171	Add error messages for user authentication (frontend)	Task		DONE	AS

## 4.2. Sprint 4 (8 Mar - 22 Mar)

During Sprint 4, the frontend development began with the user dashboard design (to allow users to access profile settings, history, templates for summarization and API access). A word count feature was implemented in the summarizer component, restricting usage for texts below 125 words unless users opt for premium features with the creation of an account. Backend enhancements for YouTube video summarization included audio extraction using Amazon services. The User Interface branch was merged with the main branch, solidifying backend-frontend integration. While some error messages and user dashboard functionalities are still under development, functionalities for storing usernames in the database and identifying logged-in users for premium

feature access have been successfully added. Test cases for account authentication were also incorporated.

#### Completed issues

Key :	Summary :	Issue type :	Epic :	Status :	Assignee
SCRUM-179	Add handling for if an inputted email is not valid	Task		DONE	AA
SCRUM-178	Add handling for if an inputted link is not valid	Task		DONE	AA
SCRUM-11	As a pro user, I want the ability to reset my password so that I have an alternative in c...	Story	USER DASHBOARD	DONE	AA
SCRUM-17	As a pro user, I want to log in to or log out of the tool so that I can access personalize...	Story	USER DASHBOARD	DONE	AA
SCRUM-28	As a pro user, I want to be able to delete my account so that all my data is deleted.	Story	USER DASHBOARD	DONE	AA
SCRUM-158	Ensure Reviews component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-150	Ensure Login component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-99	Create History component for user dashboard	Task		DONE	AS
SCRUM-181	Connect the front-end and back-end user authentication functionality.	Task		DONE	AA
SCRUM-166	Integrate a word limit for the text summarization box	Task		DONE	MB
SCRUM-155	Ensure VerifyUser component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-152	Ensure Signup component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-151	Ensure Password component responsiveness for all screen sizes	Task		DONE	MN
SCRUM-153	Ensure Summarizer component responsiveness for all screen sizes	Task		DONE	MB
SCRUM-182	Integrate the summarization functionality and the front end	Task		DONE	MS
SCRUM-12	As a pro user, I want the ability to change my profile settings (username, email) so that I have...	Story	USER DASHBOARD	DONE	AA
SCRUM-183	Add hide/unhide functionality for password inputs	Task		DONE	AS
SCRUM-184	Create change email page	Task		DONE	AS
SCRUM-185	Create delete account page	Task		DONE	AS
SCRUM-133	Add a link to URL shortener in the summarizer component	Task		DONE	MB
SCRUM-160	Add an export button next to summarized content	Task		DONE	MB
SCRUM-71	Create a word count input to allow custom text summarization	Task		DONE	MB
SCRUM-84	Add functionality to edit and delete user history	Task		DONE	HS
SCRUM-16	As a pro user, I want to view my tool's history so that I can access previous queries.	Story	USER DASHBOARD	DONE	HS
SCRUM-27	As a pro user, I want to be able to edit, or delete my tool's history for privacy reasons.	Story	USER DASHBOARD	DONE	HS
SCRUM-74	Add a history section to view previous queries	Task		DONE	HS
SCRUM-164	Add functionality for pro users to view the click counts on their link.	Task		DONE	AA
SCRUM-60	As a pro user, I want to see the activity (how many times clicked) on the shortened link that I ...	Story	CUSTOM SUMMARIZATION	DONE	AA
SCRUM-187	Add handling for if a user inputs a link that is valid but does not begin with http or https	Task	URL SHORTENING	DONE	AA
SCRUM-188	Ensure that the system is able to recognize and handle whether a user is logged in or not log...	Task	USER DASHBOARD	DONE	AA
SCRUM-189	Add functionality to allow users that are logged in to be associated with their shortened link ...	Task	URL SHORTENING	DONE	AA
SCRUM-191	Add a name text field to the create account page and make sure it is responsiveness	Task		DONE	MN
SCRUM-194	Add password requirements for create account and reset password page	Task		DONE	MN

## 5. Problems

### 5.1. User login issue when page refreshes

In the early development of the user authentication and user management, we were able to add information to the database with ease, however, there was an initial challenge in trying to figure out how to represent a user being 'logged in' or 'logged out' to the front end. Initially, we opted to use AuthContext from React, which worked, so long as you never refreshed the page. The moment the page is refreshed, the system would 'forget' that there was a user logged in. This is not ideal. So, we opted to use local storage caching to represent users being logged in, and various flags in the system such as their login method, or their name. This cache would then be deleted when the user logs out or deletes their account. Currently, it will persist even after inactivity or when the page is closed for hours, and we are investigating a 'time-out' feature to simulate other websites with this functionality.

### 5.2. URL will not shorten without HTTP

In testing the URL shortener, we noticed that if a user simply inputs [www.website.com](http://www.website.com) without including https:// or http:// as a prefix, the shortened link that they use would fail to redirect. This workaround for this issue was to manually add the prefix http:// before a website if it does not already include http:// before it. This error seemed to be due to the restrictions or limitations of Python's redirect function, as it cannot handle worldwide web prefixes and requires the hypertext transfer protocol prefix.

## 6. Future Steps

### 6.1. Frontend

- Implement dark mode for the entire website.
- Make sure the user dashboard and summarizer are responsive for all screen sizes.
- Ensure all required pages exist and can be linked to.
- Add more UI feedback, such as error messages and loading spinners during API fetching processes.
- Ensure all premium feature components are only available when the user is logged in.

## 6.2. Backend

- Add additional test cases, especially for the main functions like the Summarizer
- Further, integrate the backend into the frontend
- Add functionality for users to access our tool's API
- Look into deploying and hosting the website on a server
- Introduce user testing of the website after deployment

## 7. Meeting Notes

[All Meeting Notes](#)