CS491 Senior Design Project

Analysis and Requirement Report

ContentGuard

Project Specification Document

T2321

Zeynep Derin Dedeler, Burak Öztürk, İlayda Zehra Yılmaz,

Bengisu Buket Kardoğan, Gülin Çetinus

Supervisor: Ayşegül Dündar

Innovation Expert: Cem Çimenbiçer

Dec. 8, 2023

1. Introduction

In an era dominated by the digital realm, social media platforms have become integral agents for information exchange. Among these platforms, Twitter stands out as a dynamic space characterized by its nonstop updates and diverse content. As individuals dive into the world of Twitter, they face the following challenges: navigating the expansive array of content effectively and managing the time devoted to engaging with Twitter.

The widespread use of social media, especially on platforms such as Twitter, has given rise to a prevalent issue: the unintentional dissipation of time on topics that may not align with users' primary interests or goals. As individuals traverse the endless stream of tweets, the risk of diverting attention to unrelated or unproductive content becomes increasingly apparent. In this setting, our senior design project evolves with the fundamental objective of empowering users to shape their social media experience.

The project arises from a modern challenge of finding the balance between the irresistible appeal of social media and the need for effective time management. With platforms providing an increasing variety of content, users frequently feel overwhelmed by a flood of information. Therefore, there's a need for a solution that helps navigate content efficiently and ensures control over digital interactions.

Recognizing the widespread nature of this challenge, our project aims to meet a fundamental requirement: the capability to regulate and personalize the Twitter experience. Users need a tool that goes beyond passive observation. They require a proactive solution that assists in actively managing their digital engagement. The goal is not merely to monitor user activity but to provide features that empower individuals to create a purposeful and intentional journey on Twitter.

As we explore the details of our project, the ContentGuard, our goal is to encourage innovation, user-centric design, and a dedication to transforming the dynamics of social media interaction. By fostering a mutually beneficial relationship

between users and their Twitter experience, our project aims to initiate a new era of mindful digital engagement.

2. Current System

In the contemporary digital landscape dominated by social media, particularly platforms like Twitter, users grapple with challenges related to information overload and time management. The pervasive use of social media has given rise to a common issue: the unintentional dissipation of time on content that may not align with users' primary interests or goals. As individuals navigate the dynamic space of Twitter, the risk of diverting attention to irrelevant or unproductive content becomes increasingly apparent.

Existing systems in social media management primarily focus on passive observation, lacking the proactive features necessary for effective time management and personalized content curation. While some tools offer basic content filtering options, they often fall short in allowing users to actively shape and regulate their digital interactions. The need for a solution consolidating content navigation and empowering users to control and personalize their social media experience is evident.

Several tools attempt to address content filtering and time management aspects on social media platforms. Nevertheless, most of these tools exhibit limitations in functionality, falling short of providing a holistic solution to address users' challenges. Frequently, these solutions lack the innovation required to adapt to the dynamic nature of social media dynamics and the expanding array of content.

Our senior design project, the ContentGuard, emerges as a response to this modern challenge. It distinguishes itself from existing systems by offering a proactive solution that actively involves users in managing their digital engagement on Twitter. Unlike passive observation tools, our project goes beyond mere monitoring, providing features that empower individuals to create a purposeful and intentional journey on Twitter.

Key differentiators include advanced content filtering algorithms, real-time analytics, and a user-centric design aimed at fostering a mindful digital engagement experience. ContentGuard not only addresses the current shortcomings of existing systems but also sets itself apart by encouraging innovation and a dedication to transforming the dynamics of social media interaction. By establishing a mutually beneficial relationship between users and their Twitter experience, our project aims to usher in a new era of thoughtful and personalized digital engagement.

3. Proposed System

3.1 Overview

In this project, ContentGuard, we plan to develop a Chrome extension that will help users track and categorize their Twitter content based on individual interests, allowing for personalized and focused experiences while also empowering users with control over time management and content filtering for a purposeful and efficient social media interaction.

3.2 Functional Requirements

- The user can register to the application and sign in.
- The user can activate the extension by clicking the activation button on the processing page content.
- The user can deactivate the extension by removing it, disabling it, or signing out.
- The user can delete their account.
- The user can set the filtering options for multiple content types.
- The extension can filter tweets from the tweet feed according to the user's filtering settings.
- The user can determine timer constraints for filtering content types.
- The extension will track the time spent on consuming different content types and be able to create a report and display it to the user.
- The extension will activate chosen filters when timer constraints are met for these content types.
- The user can select specific keywords or hashtags for filtering.

- The extension will remove a tweet that includes chosen keywords or hashtags from the user tweet feed.

3.3 Nonfunctional Requirements

3.3.1. Usability

- The user interface should be clearly understood by the users and needs to not have a learning curve to ease the users into using it without worrying about the complexity of the user interface.

3.3.2. Reliability

- The project should be able to accurately categorize each interacted tweet's content.
- The project's servers should not be down for a long amount of time, which affects the users' experience.
- When an error occurs, the program needs to inform the user that an error occurred and if they want to report the error to us developers.

3.3.3. Performance

- The content tracker part of the project should be lightweight and not impact the user experience by slowing down the load time on Twitter.
- The report part of the project should not take too much time that users are turned down to use the extension, which is around 5-10 seconds at most.

3.3.4. Extensibility

- The project should be open to extensibility via adding new features to the project according to the feedback we are getting from instructors and the innovation expert.

3.3.5. Scalability

- Content Tracker will be introduced to Twitter first, but after completion of the Twitter extension, the project can be extended to other platforms such as Instagram, Youtube, Bluesky, etc.

3.4 Pseudo Requirements

3.4.1. Implementation Requirements

- Twitter API or web scraping will be used to gather training data for the AI model.
- Datasets that are available on the internet can be used as well.
- Github is used for version control. It allows parallel working, cloud storage for code and code-related files, and a running environment with Github Pages (for just the info website for now).
- Chrome browser is used as the testing environment for the extension.
- AI models will be trained with Google Cloud servers and stored in the Firebase servers.
- User data is planned to be stored in a non-relational database such as MongoDB.
- A project tracker like Github Issues is to be used to plan the development process.

3.4.2. Economic Requirements

- Twitter API allows for pulling 10,000 tweets for $100/month and 1,000,000 tweets for $5;000/month. Hence, web scraping is the more feasible option.[1]
- Datasets that can be found online are usually free.
- The part of the Github functionality we will use is free.
- Chrome browser and its extension functionality for the Chromium-based browsers are free.
- Google services have special pricing models for educational purposes, making them feasible for this project.
- MongoDB is open-sourced and free.

3.4.3. Ethical Requirements

- The data we will get by processing the feed and users' information are confidential and won't be shared by third parties.
- The data will be stored in a secure place to protect user privacy.
- The data will not be stored so that its source can be linked.
- The data will be analyzed and grouped to minimize biases in the subject, and experts may be consulted for determination.

- The Ethics of the National Society of Professional Engineers will be followed by our team in this project [2].
- If some features are decided to be paid features or the subscription is added to the Twitter Content Tracker in the future by the development team, it is understood by us that the payments system should be secure, and the team will make their efforts to make it as such.

3.5 System Models

3.5.1 Scenarios

Use Case #1

Use case name: Get report

Participating actors: User

Flow of events:

1- The user clicks the UI component that allows them to get their current session report.

2- The system creates a report for the user on their current session information.

3- The download finishes on the user's computer.

Entry conditions:

The user has to be on the get report page and click the corresponding UI component.

Exit conditions:

The download operation for the report is finished.

Use Case #2

Use case name: Set topic to be filtered from homepage

Participating actor: User

Flow of events:

1- The user goes to the filters page.

2- The user chooses the UI component to add a topic filter.

3- The system receives the information about the topic filter and creates it.

Entry conditions:

The user has to be on the filters page and click the corresponding UI component.

Exit conditions:

The filter already exists or the filter is added successfully.

Use Case #3

Use case name: Set hashtag/keyword to be filtered from the homepage

Participating actor: User

Flow of events:

1- The user goes to the filters page.

2- The user chooses the UI component to add a keyword filter.

3- The system receives the information about the keyword filter and creates it.

Entry conditions:

The user has to be on the filters page and click the corresponding UI component.

Exit conditions:

The filter already exists or the filter is added successfully.

Use Case #4

Use case name: View filters

Participating actor: User

Flow of events:

1- The user goes to the filters page.

2- The system shows all the filters that the user has added.

Entry conditions:

The user has to be on the filters page.

Exit conditions:

None.

Use Case #5

Use case name: Update filter

Participating actor: User

Flow of events:

1- The user goes to the filters page.

2- The user clicks on one of the existing filters.

3- The user edits the filter.

4- The system updates the filter accordingly.

Entry conditions:

The user has to be in the filters page and there has to be an existing filter.

Exit conditions:

There is no existing filter or the filter is updated successfully.

Use Case #6

Use case name: Set timer

Participating actor: User

Flow of events:

1- The user goes to the timers page.

2- The user clicks the UI component to add a timer.

3- The user chooses a filter to add a timer.

4- The user sends information about the timer to the system.

5- The system creates the timer.

Entry conditions:

The user has to be on the timers page and there has to be an existing filter.

Exit conditions:

There is no filter.

Use Case #7

Use case name: View timers

Participating actor: User

Flow of events:

1- The user goes to the timers page.

2- The system shows all the timers that the user has added.

Entry conditions:

The user has to be on the timers page.

Exit conditions:

None.

Use Case #8

Use case name: Update timer

Participating actor: User

Flow of events:

1- The user goes to the timers page.

2- The user clicks on one of the existing timers.

3- The user edits the timer.

4- The system updates the timer accordingly.

Entry conditions:

The user has to be on the timers page and there has to be an existing timer.

Exit conditions:

There is no existing timer or the timer is updated successfully.

Use Case #9

Use case name: Remove filter

Participating actor: User

Flow of events:

1- The user goes to the filters page.

2- The user clicks on the UI component to remove a filter.

3- The system removes the filter.

Entry conditions:

The user has to be on the filters page and there has to be an existing filter.

Exit conditions:

There is no filter or the filter is removed successfully.

Use Case #10

Use case name: Remove timer

Participating actor: User

Flow of events:

1- The user goes to the timers page.

2- The user clicks on the UI component to remove a timer.

3- The system removes the timer.

Entry conditions:

The user has to be on the timers page and there has to be an existing timer.

Exit conditions:

There is no timer or the timer is removed successfully.

Use Case #11

Use case name: Create report for user

Participating actor: System

Flow of events:

1- The user initiates a report query.

2- The system creates a report according to the user's analytics.

3- The system sends the report for the user to be downloaded.

Entry conditions:

The user has to make a query to create a report.

Exit conditions:

The report is created and sent to the user.

Use Case #12

Use case name: Create filter

Participating actor: System

Flow of events:

1- The user creates a filter.

2- The system creates a filter according to the user's input.

Entry conditions:

The user has to make a query to create a filter.

Exit conditions:

The filter is created correctly.

Use Case #13

Use case name: Check authentication

Participating actor: System

Flow of events:

1- The system checks if the user is logged in with their Twitter account.

Entry conditions:

None.

Exit conditions:

The user is not logged in.

Use Case #14

Use case name: Create timer

Participating actor: System

Flow of events:

1- The user creates a timer.

2- The system creates a timer according to the user's input.

Entry conditions:

The user has to make a query to create a timer.

Exit conditions:

The timer is created correctly.

3.5.2 Use Case Model

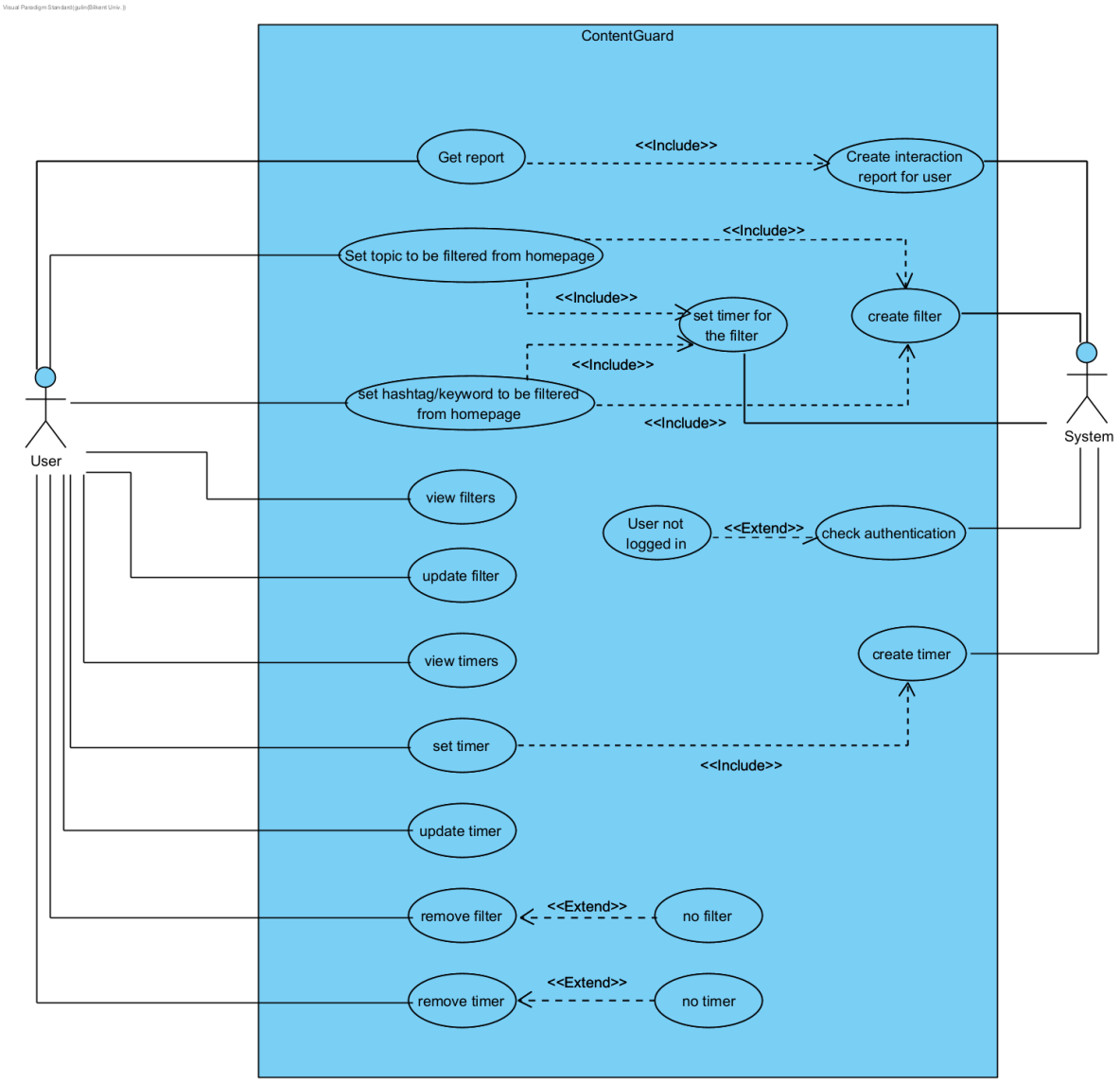

Figure 1. Use case model of User and System.

3.5.3 Object and Class Model

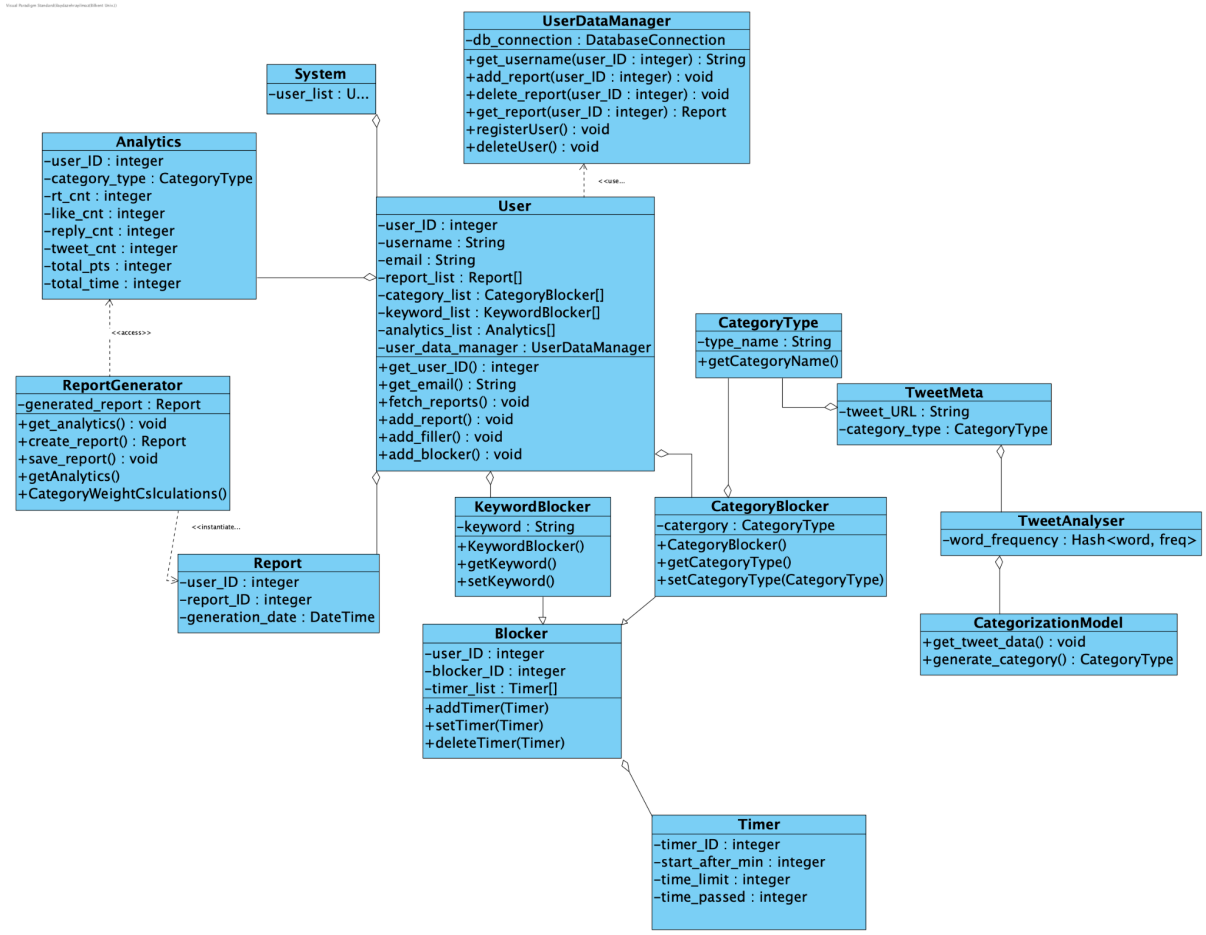

Figure 2. Object and class model of User and System.

System: The system holds all the users in the program.

User: Holds user information and modifies the information.

UserDataManager: Acts as a middle class between the database and the user class. Performs queries according to the User class's modification.

CategoryType: Category name that is created after tweet analysis by ML model.

Analytics: Holds the information about each category type. Holds retweet, like, tweet, and reply count and time spent on per user per category. Computes a point from retweet, like, tweet, and reply count.

ReportGenerator: Creates a report according to the Analytics class for the user.

Report: Includes the time spent on each category, point distributions and the date that it's created.

Blocker: Wrapper class for KeywordBlocker and CategoryBlocker

KeywordBlocker: Blocks a keyword that is inputted by the user.

CategoryBlocker: Blocks a category that is inputted by the user.

Timer: Timer for a Blocker class. Users can choose a time for the timer to start and the timer limit.

TweetMeta: Holds the tweet information such as the tweet URL.

TweetAnalyser: Puts each word in a tweet and its frequency into a hash.

CategorizationModel: The trained ML model determines the category of the tweet

3.5.4 Dynamic Models

3.5.4.1 Activity Diagrams

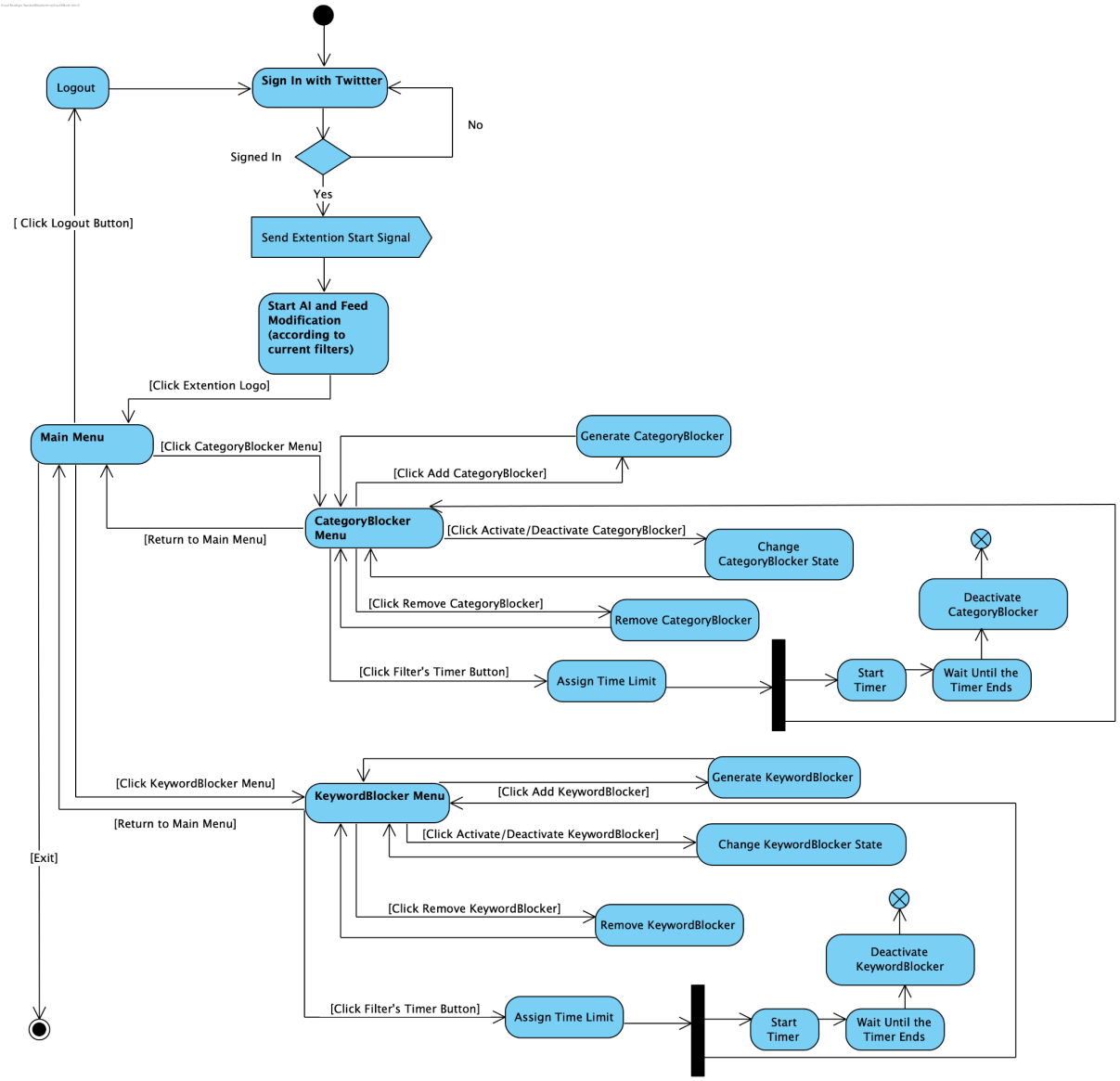

Figure 3. Overall System Activity Diagram

The activity in the Twitter extension we will create starts with signing in using Twitter to sign in to the system. If sign-in is not successful then it returns to the sign-in activity again. If the user is successfully signed in, the extension start signal is created and sent so that the already established mechanism can start working (like filtering and such) and the AI can analyze the feed and do modifications or improve itself. When the extension is clicked in somewhere after these activities the Main Menu of the extension is opened. In this menu, there are several menus like the CategoryBlocker and KeywordBlocker Menus. Both Menus use similar activities, so this will explain just one as the other only differentiates in the Keyword or Category Name in this activity diagram. When the CategoryBlocker Menu is clicked the CategoryBlocker Menu opens. If the Add CategoryBlocker Menu is clicked a new CategoryBlocker is created to be used to filter the content. When the Activate/Deactivate is clicked the Blocker's state is changed and either activated or deactivated. There is also an activity that can be reached by clicking remove CategoryBlocker and this removes the particular blocker from the system. The Assign Time Limit activity can be reached for each CategoryBlocker by clicking that Filter's Timer Button. This will assign a time limit to that blocker and Start Timer in the background while the user returns to the CategoryBlocker Menu. In the background, the timer counts, and when it finishes the CategoryBlocker is activated and the feed is filtered in that category. Then this background flow ends. These procedures are the same for the KeywordBlocker except it does not block the category but that special keyword so it just does keyword filtering activities instead of AI finding and filtering a subject. Finally, users can click Log Out and Log Out by using the Main Menu. The "Exit" ing is the finishing of the activity and how the extension stops working.

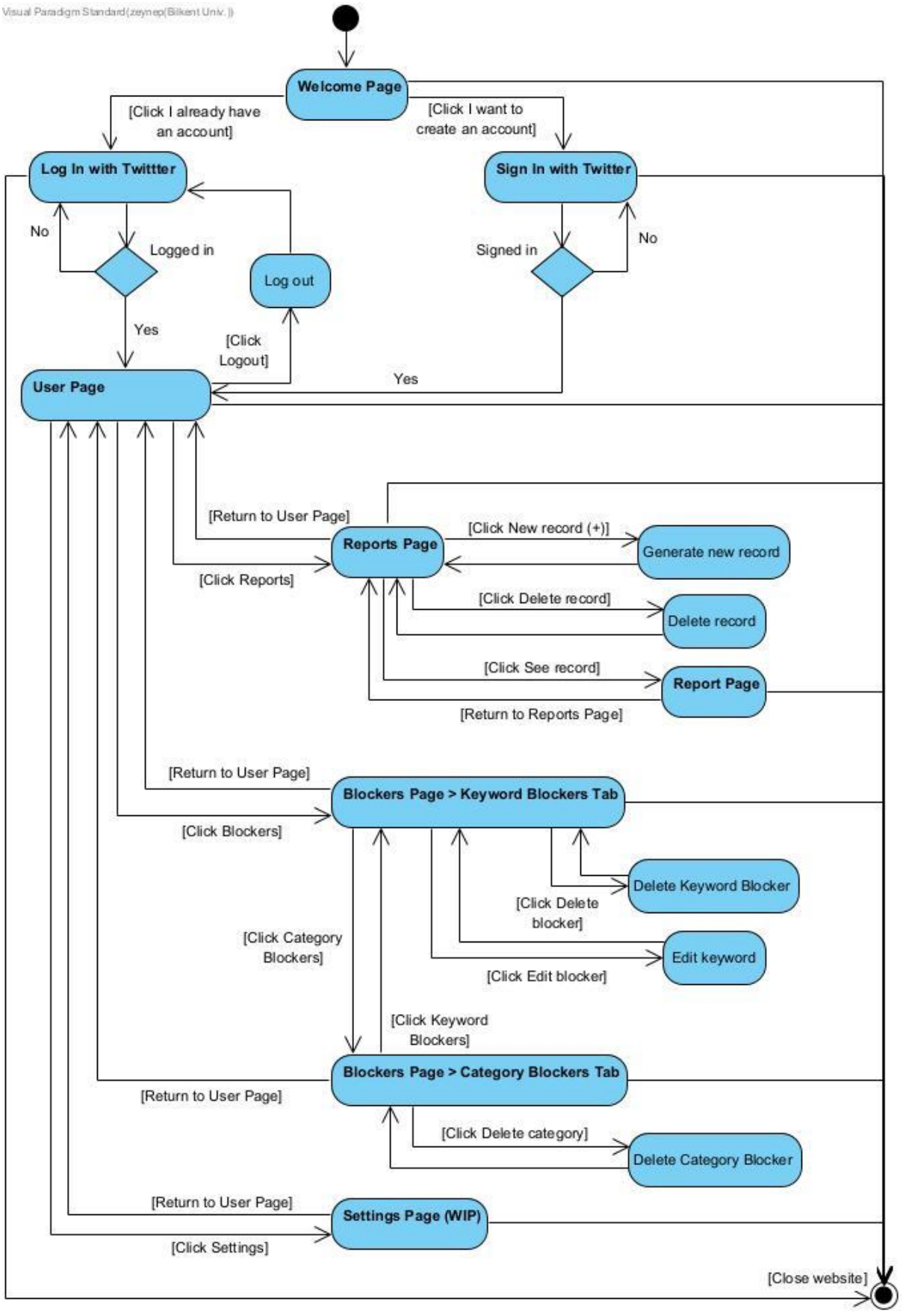

Figure 4. Web Page Activity Diagram

The Welcome page is the first page that welcomes the user at the start. It has two functional components, “I already have an account” and “I want to create an account” buttons. Both of them use Twitter authorization API but while prior signs in into an already created account, later creates an account for the new user.

After logging in, users get to the User Page. There are buttons that navigate the user to the Reports, Blockers, and Settings pages. Those buttons, the user’s Twitter name, and a Logout button are all in a sidebar component, therefore they are available from all pages.

Reports page shows the user’s saved reports and there are buttons for opening the detailed view of the report or deleting the report for each report. Also there is the “Create new record” button which is denoted by the (+) sign. It allows the creation of a new report for the user.

The report page does not have any interactive components of its own.

Blockers page has two tabs for two types of blockers: Keyword blockers and Category blockers. Both have the same structure. They show their respective blocker types as lists and the Keyword Blockers tab has edit and delete functions for each Keyword Blocker while the Category Blockers tab only has a delete function.

The settings page is a WIP as shown in the diagram. We plan to fill it with the user-related settings that we find needed in the development process.

Two details are left out of the diagram for readability. Logout is available from all the pages since it is in the sidebar and all pages are available from all pages thanks again to the sidebar.

The final node is reached by closing the website and it is reached from all the pages because of this structure.

3.5.4.2 State Diagrams

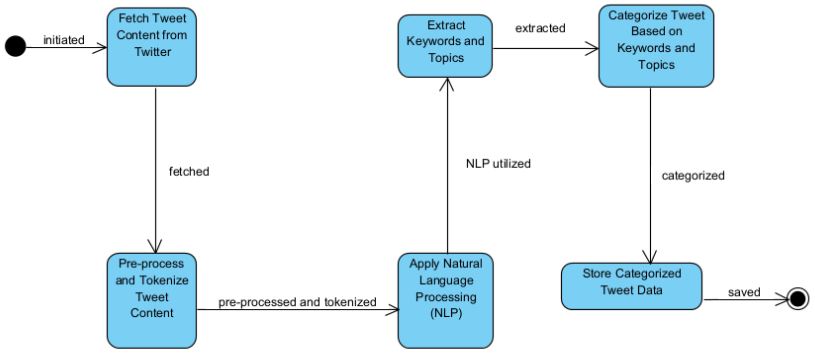

Figure 5. Categorize Tweets State Diagram

Figure 5 illustrates the stages of processing a tweet for the extension. The extension begins to categorize the tweets with the "Fetch Tweet Content from Twitter" state, initiated by up on sing-ining on to the extension. Once the tweet is fetched, the state transitions to "Pre-process and Tokenize Tweet Content", where the tweet is prepared for analysis. The next stage, "Apply Natural Language Processing (NLP)", involves using NLP techniques likely to extract meaningful keywords and topics. Once the extraction is done, the tweet moves to the "Categorize Tweet Based on Keywords and Topics" state, where it is classified according to the identified keywords and topics. The final state is "Store Categorized Tweet Data", where the now categorized tweet data is saved to a storage system. The process completes when the data is successfully stored.

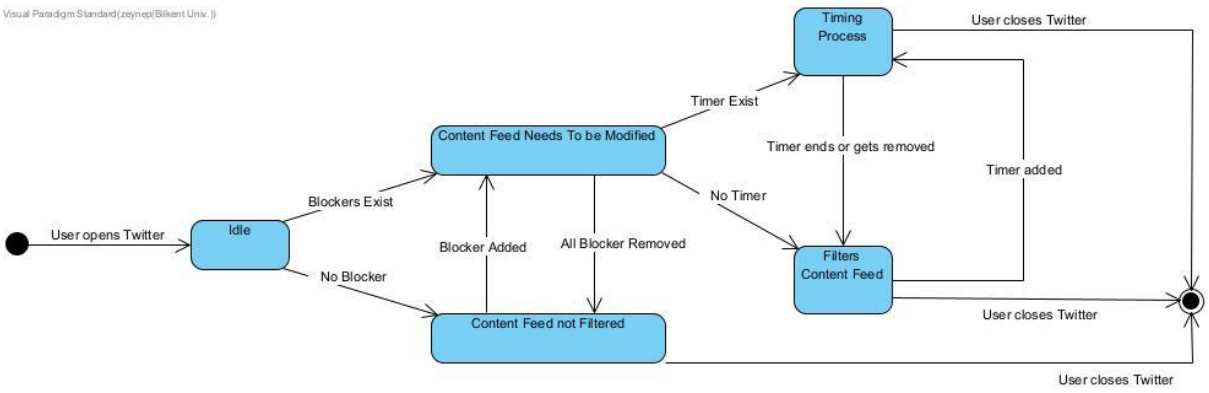

Figure 6. Blocker Mechanism State Diagram

As the user logs in to Twitter and logs in to the extension it is in its idle state. From this idle state, if there exists any Blockers it goes on to the state of modifying the Twitter feed accordingly, if the content feed is displayed to the user without any modifications. The Twitter feed is filtered accordingly when a blocker is added to the extension. And if every blocker is removed from the extension, the content feed returns to be displayed as before. When a Blocker exists and has a Timer, the timing process starts, and when the determined time is up, only then will the blocker be activated to filter the Twitter feed.

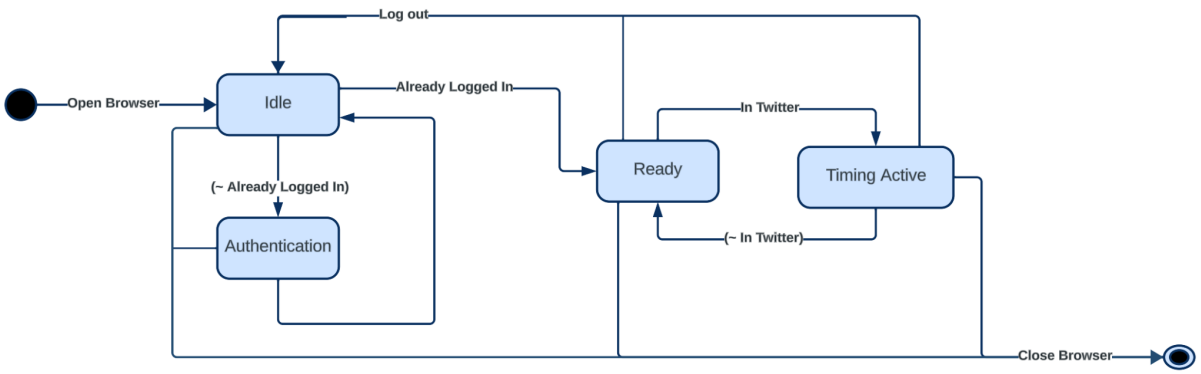

Figure 7. Timing Mechanism State Diagram

The timer state starts with the Idle state when the browser is opened. When the user logs in and the authentication is completed, the timer state gets into the Ready state. In the Ready state, the timer can be started when Twitter is opened, and necessary actions are taken. So when these actions are done, the timer state becomes Timing Active. When the user is not on Twitter, the timing is not counted, and the timer state is ready again. When the user logs out, the timing stops being active and returns to the idle state, where it is not ready and not active. If the authentication is not done, the state turns into Idle again. In any of these states, if the browser is closed, then the timing states reach their final.

3.5.4.3 Sequence Diagrams

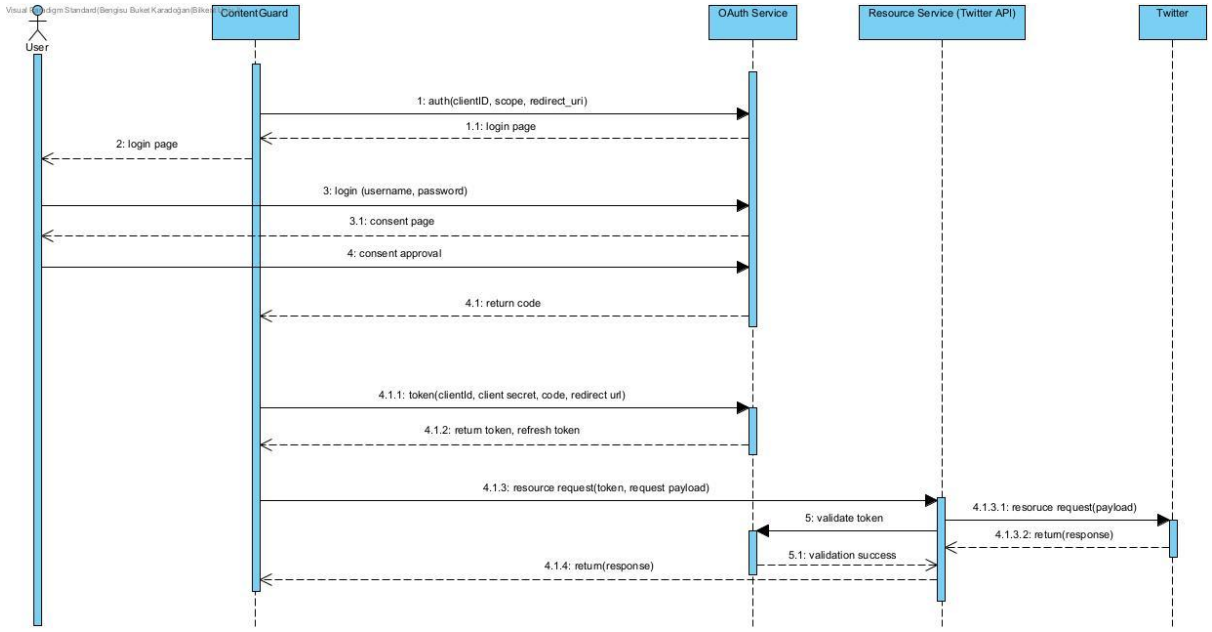

Figure 8. Verify Account Sequence Diagram

The user initiates the process by requesting authorization from the OAuth service through the ContentGuard and trying to access a protected resource that requires authentication. The ContentGuard sends a request that includes its client ID, the scope of access, and a redirect URI to the OAuth service. The OAuth service responds by presenting a login page to the user. The user interacts with the login page by entering their username and password. If the username and password are correct, the user is presented with a consent page asking for approval of the requested access scope. The user approves the requested consent on the consent page. The OAuth service generates an authorization code and sends it back to the ContentGuard application. The ContentGuard makes a request to exchange the authorization code for an access token. This request includes the client ID, client secret, the previously received authorization code, and the redirect URI. The OAuth service refreshes and returns the token to the ContentGuard application. Then, the ContentGuard sends a resource request to the Twitter API with the refreshed token and requested payload. Twitter API with the requested payload sends a resource request to the Twitter application and sends a message to the OAuth service to validate the token received. With

the responses from the Twitter application and the OAuth service, the Twitter API sends a response to the ContentGuard application.

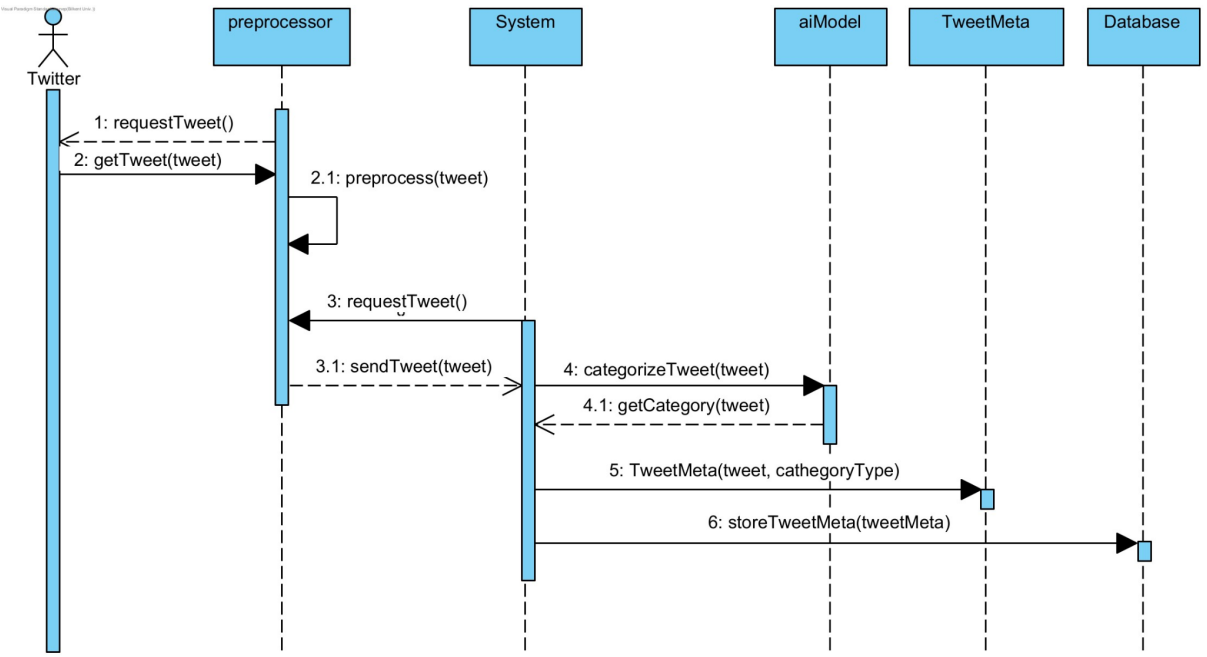

Figure 9. Process Categorize Sequence Diagram

With requestTweet(), the system sends an external request to Twitter for a tweet. Twitter responds by sending the tweet to a component labeled 'preprocessor' in the getTweet(tweet) function. In preprocess(tweet) function, the preprocessor takes the tweet and performs data cleansing operations on it. Preprocessing includes cleaning the text, removing special characters, normalizing the data, etc. Then the system takes the preprocessed tweet with requestTweet() and sendTweet(tweet) functions from the preprocessor and sends it to the aiModel with categorizeTweet(tweet) function. Then aiModel categorizes the tweet according to the meaning of the tweet, and then System takes the category information of the tweet by using the getCategory(tweet) function. Finally, the System creates a TweetMeta object to store the original tweet data determined CategoryType object and stores it in the Database with TweetMeta(tweet, categoryType) and storeTweetMeta(tweetMeta).

3.5.5 User Interface - Navigational Paths and Screen Mock-ups

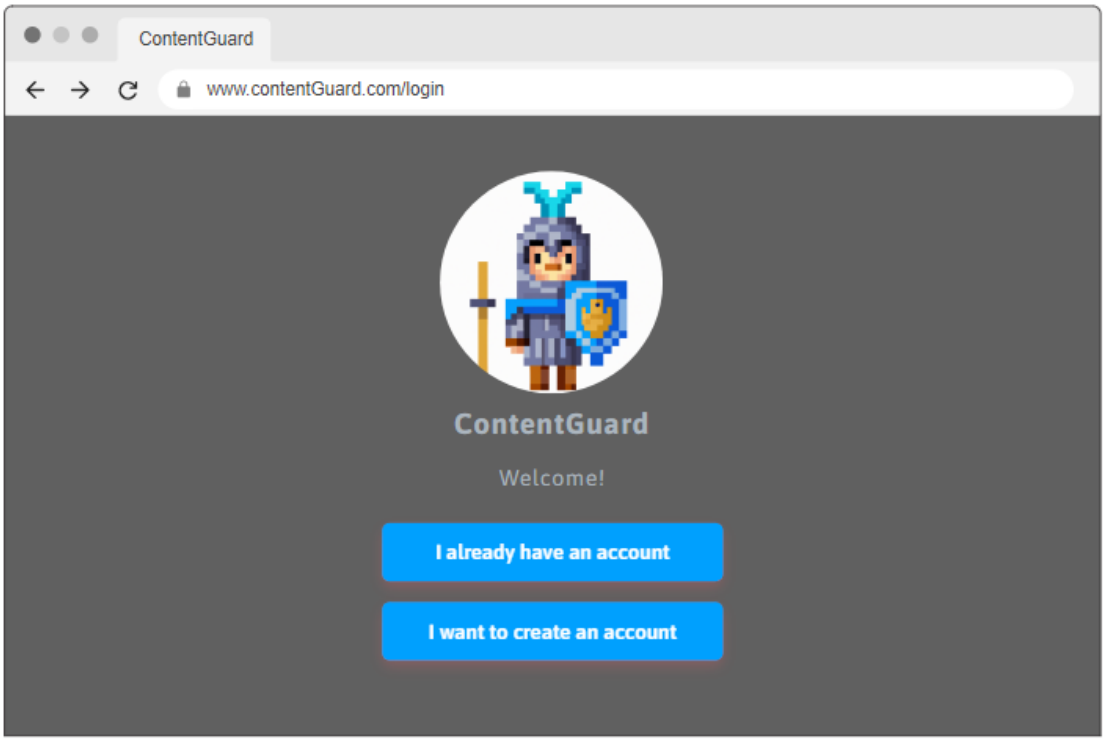

Figure 10. Login Page User Interface Mock-Up

On this page, the user can either log in to an existing account or create a new account with their Twitter account. Buttons will direct users to the Twitter authentication page.

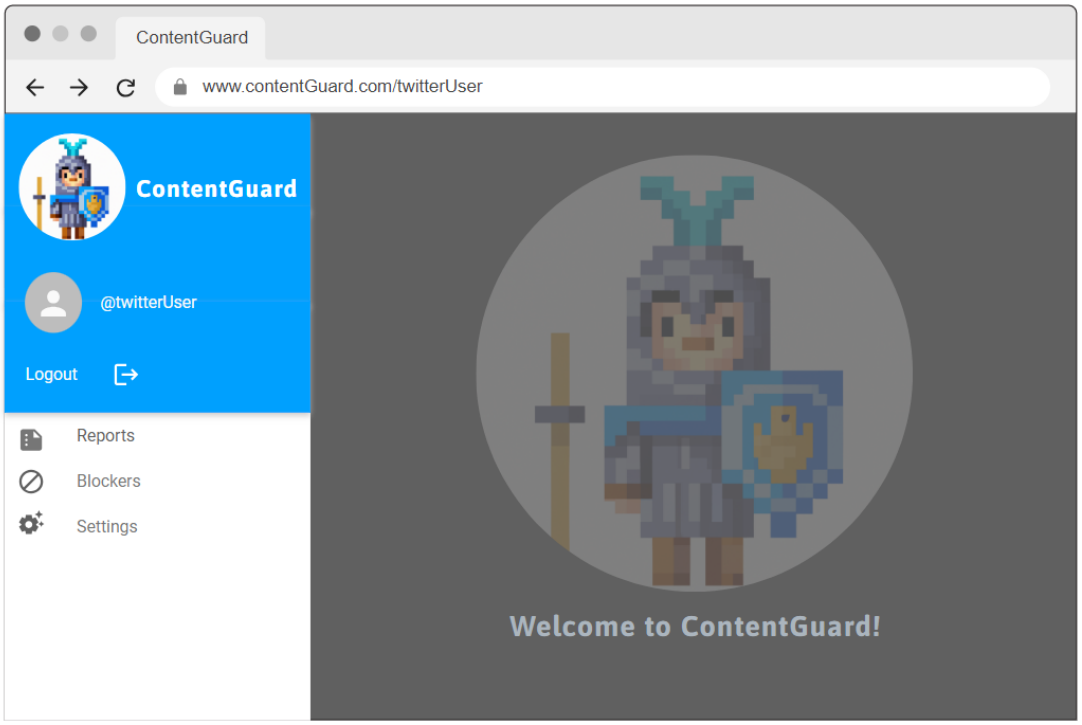

Figure 11. Home Page User Interface Mock-Up

This is the ContentGuard home page. This page comes after login. The user can see that s/he is logged in successfully. The user can navigate in the application and can logout using the side menu.

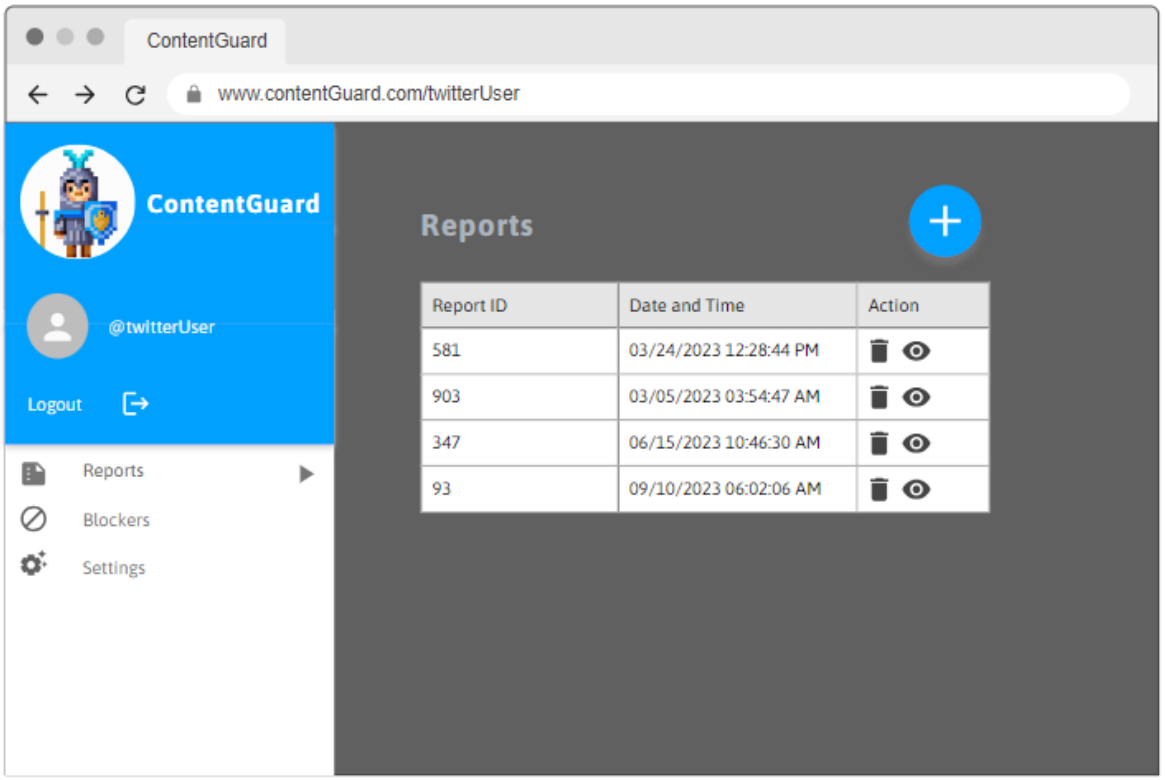

Figure 12. Home Page Reports User Interface Mock-Up

The user can access the reports by clicking the reports button on the side menu. On the reports page reports are listed. The user can create a new report by clicking the plus icon on the right top of the page. The user can delete a report and see the report details with the action buttons.

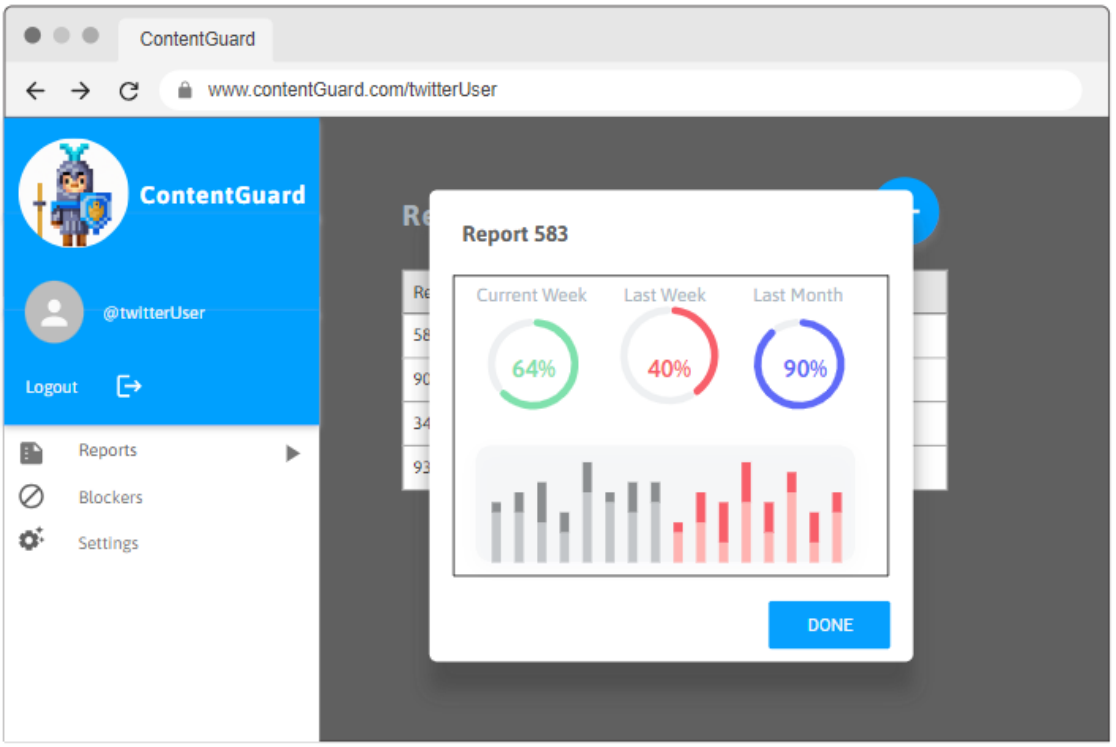

Figure 13. Home Page Reports User Interface Mock-Up

The user can access the details model for a report by clicking on the eye icon in the actions column of that report. The modal can be closed by clicking on the 'Done' button or outside of the modal.

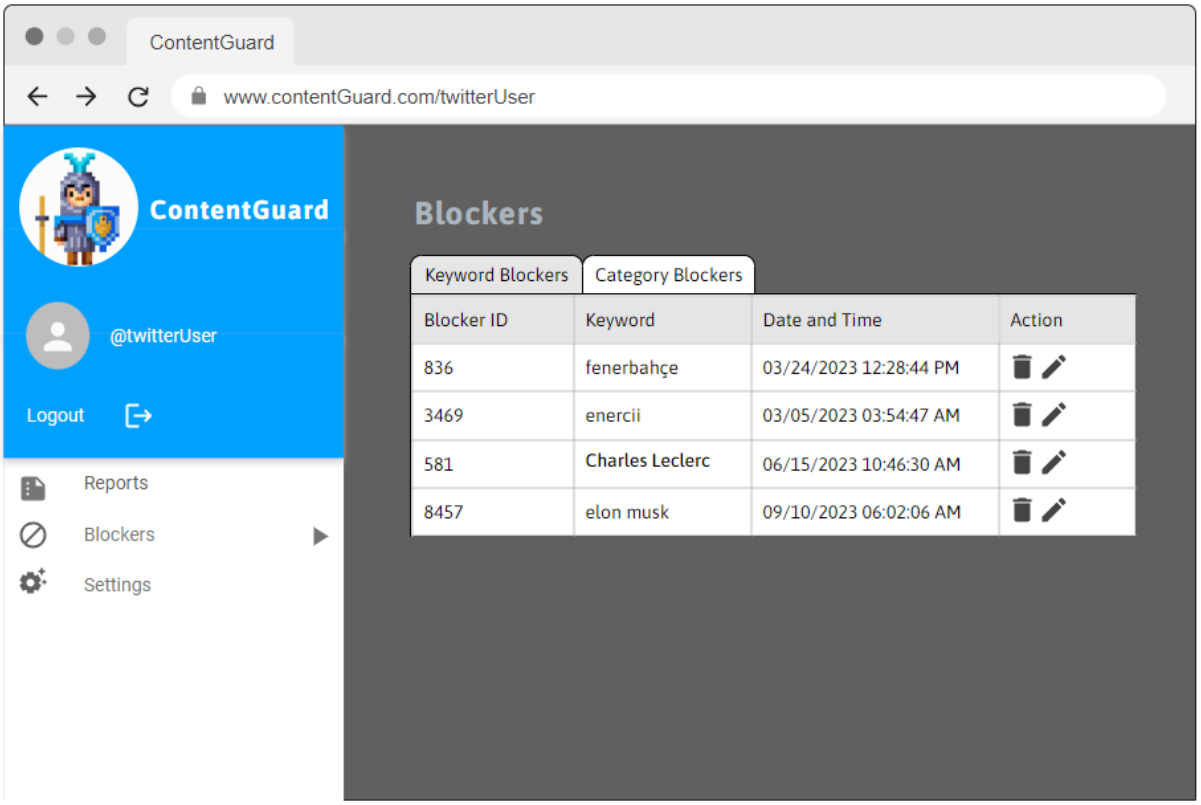

Figure 14. Blockers Page Keyword Blockers Tab Interface Mock-up

In this page, the keyword blockers of the user are listed in a table. The user can see the keyword blockers, and they can delete or change them.

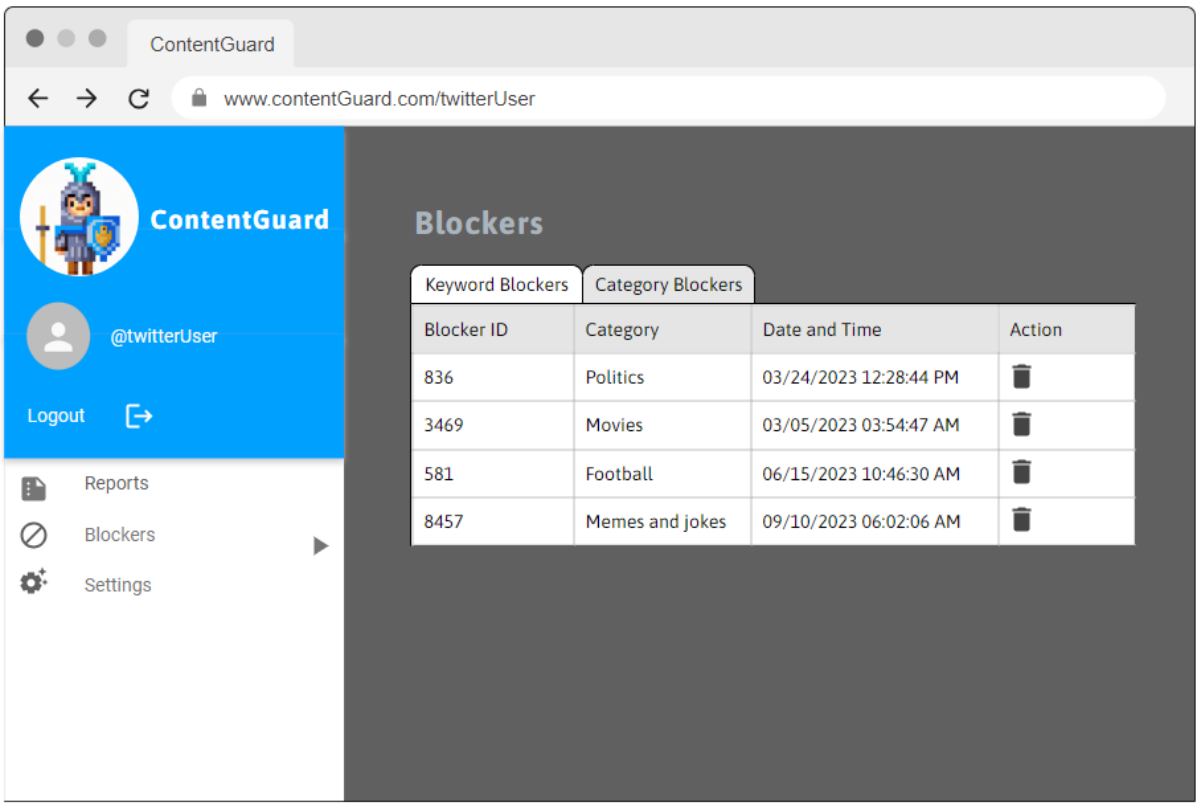

Figure 15. Blockers Page Category Blockers Tab Interface Mock-up

In this page, the category blockers of the user are listed in a table. The user can see the keyword blockers, and they can delete them.

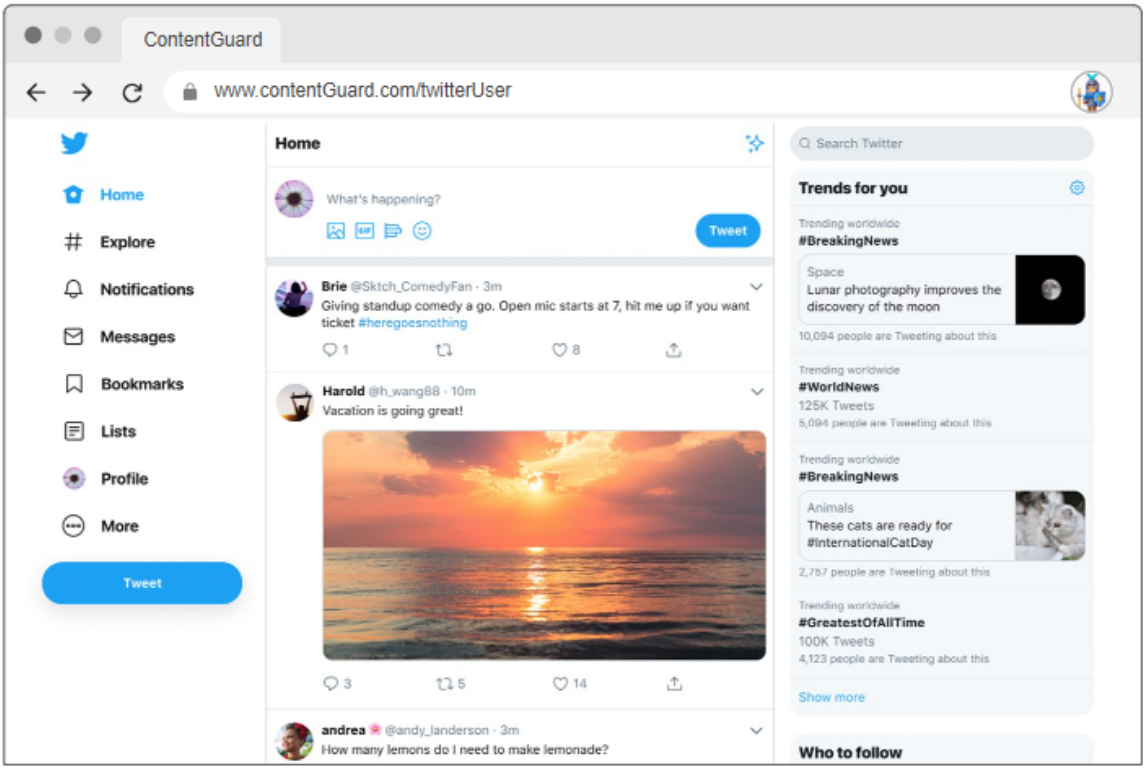

Figure 16. Extension and Twitter Page User Interface Mock-Up

The extension icon will be shown to the user with the ContentGuard icon in the browser's extension section.

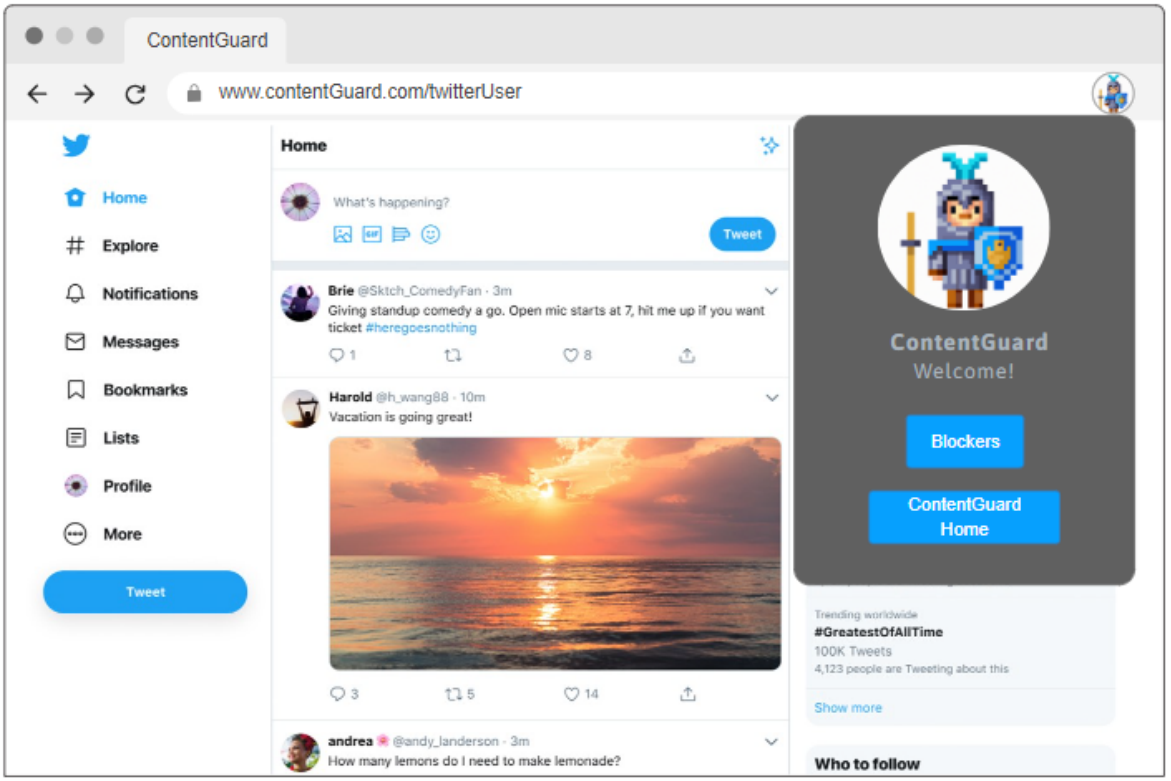

Figure 17. Extension Pop-up User Interface Mock-Up

The user can access the extension content by clicking on the icon. In this extension pop-up, there will be 2 buttons. The ‘Blockers’ button will open the blockers section in the extension pop-up and the ‘ContentGuard Home’ button will redirect the user to the ContentGuard home page.

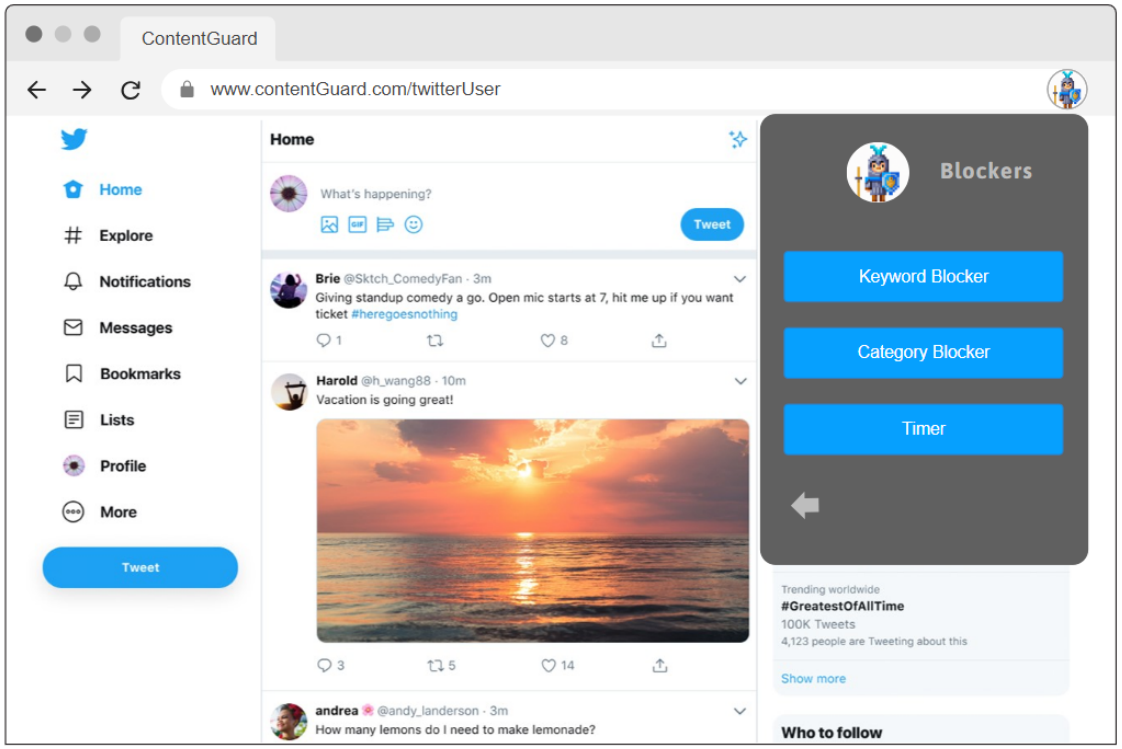

Figure 18. Extension Pop-up Main Menu User Interface Mock-Up

In the ‘Blockers’ section, the user can go to the ‘Keyword Blocker’, ‘Category Blocker’ content filtering sections, and ‘Timer’ section by clicking at the corresponding buttons. The back arrow takes the user back to the previous section.

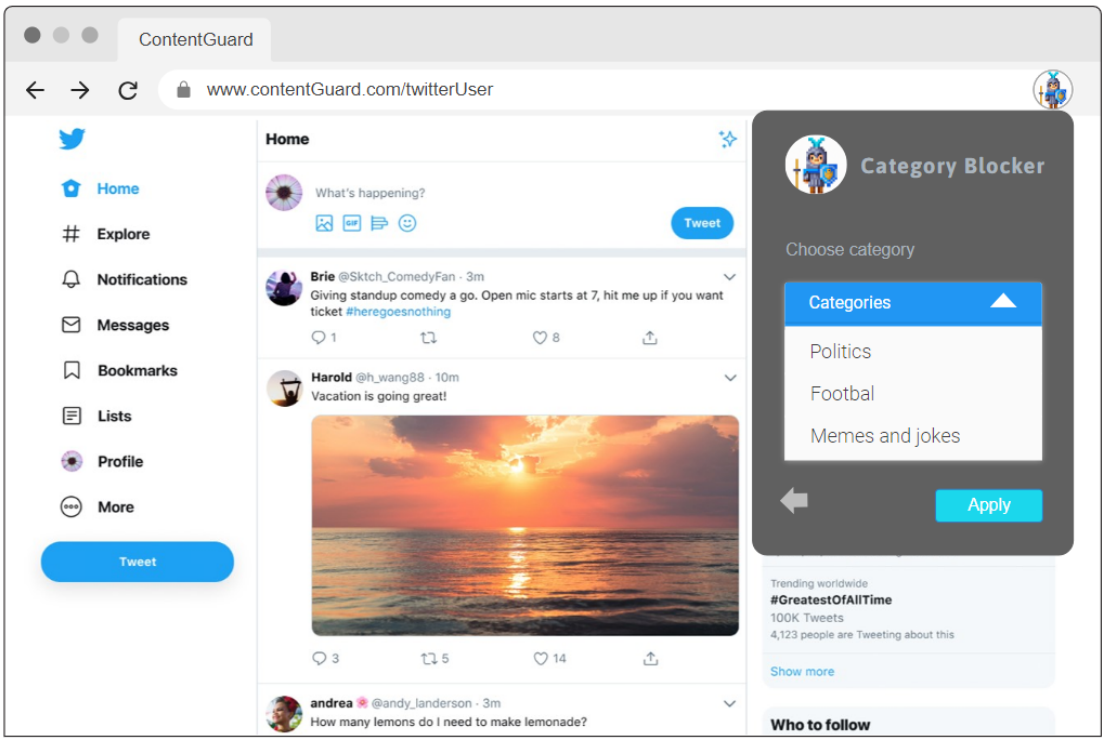

Figure 19. Extension Pop-up CategoryBlocker Menu User Interface Mock-Up

In the ‘Category Blocker’ section the user can select a category from the existing category list to be blocked with the help of the dropdown menu and apply the selection by clicking at the ‘Apply’ button. Clicking at the ‘Apply’ buttons navigates the user back to the previous section. The back arrow takes the user back to the previous section.

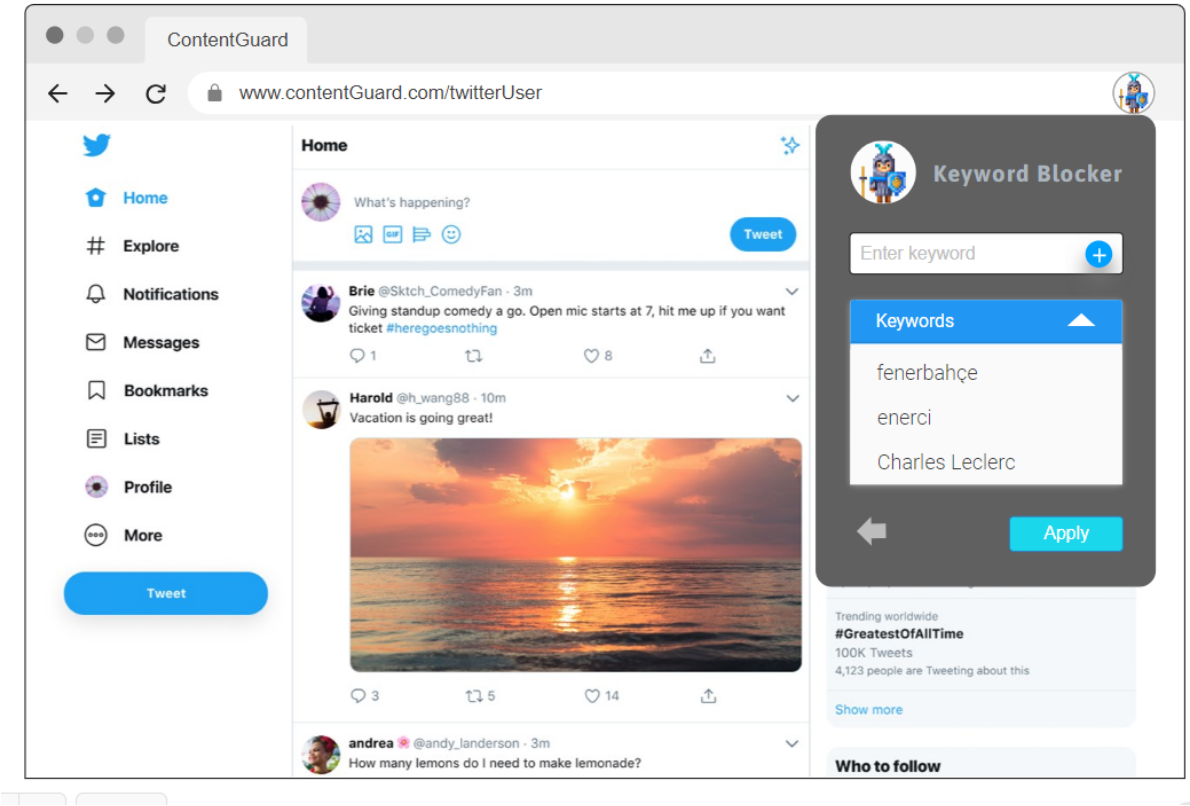

Figure 20. Extension Pop-up KeywordBlocker Menu User Interface Mock-Up

In the ‘Keyword Blocker’ section, the user can select a keyword from an existing keyword list from the dropdown menu or s/he can input a new keyword to be blocked by the text field. To add a new keyword user should click on the plus icon in the text field. By clicking on the ‘Apply’ button the user can apply his/hers selection and will be navigated back to the previous section. The back arrow takes the user back to the previous section.

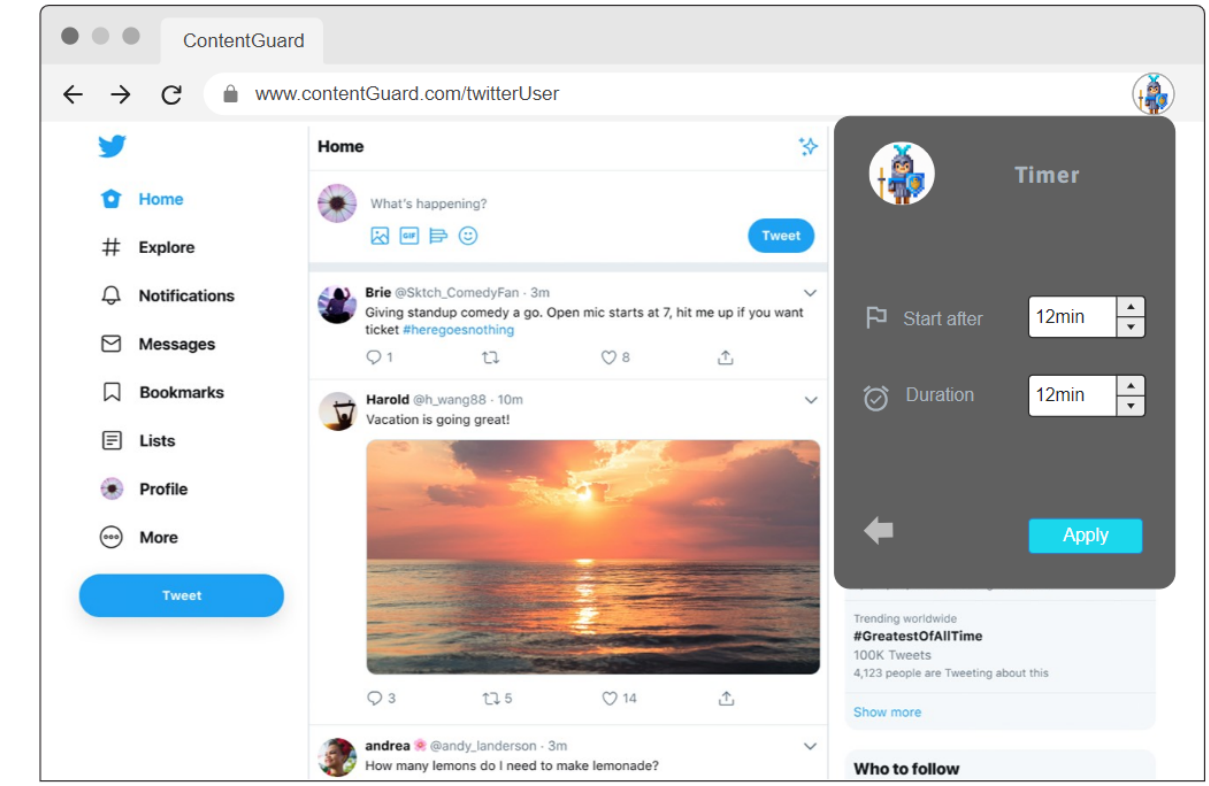

Figure 21. Extension Pop-up Timer User Interface Mock-Up

In the 'Timer' section, users can configure the properties of a timer. The section includes two input number fields. The 'Start after' field enables users to activate applied blockers after a specified time. The 'Duration' field allows users to set the duration for which the blockers will be active if any of them are applied. Alternatively, if none of the blockers are applied, it sets the time the user wishes to spend on Twitter and alerts them when the time is up. If no time is inputted in the 'Timer' section, there will be no time restrictions. The back arrow takes the user back to the previous section.

4. Other Analysis Elements

4.1. Consideration of Various Factors in Engineering Design

4.1.1 Constraints

All the constraints we considered and will consider in the future are in 8 categories: Accessibility, aesthetics, codes, functionality, interoperability, legal considerations, maintainability, and usability.

For accessibility, we plan to ensure some kind of accessibility compliance, such as WCAG. This would guide us to provide accessibility features like alternative text styles and keyboard navigation alternatives.

For aesthetics, we designed mock-up UIs that are nice to look at. While making the designs more beautiful, we also tried not to hinder usability while adjusting for aesthetics.

For codes, we aim to write clean, well-documented, and well-structured code for easy maintenance and improvement in the future. In this case, tools such as design patterns and modularization will be used when necessary.

For functionality, our main purpose in the project is to improve the Twitter experience of our users with many useful functions that are working effectively. We also pay attention to avoid spending time and effort on unnecessary features by eliminating them at the planning stage.

For interoperability, we will consider compatibility with various Chrome and Chromium versions and possible conflicts with popular extensions.

For legal considerations, we will comply with Twitter’s Terms of Service and privacy and data protection regulations. We will handle the personal data of our users ethically regardless of the law

For maintainability, we plan to make the code structure easily debuggable and updateable with the criteria in the codes part. Therefore, we will be able to do updates after the release.

For usability, we designed mock-up UIs as simplistic, self-descriptive, and easy to use as possible while keeping them pleasant. This would help to ensure a positive user experience.

4.1.2 Standards

The standard we will use for developing the project is IEEE 12207. It encompasses software life cycle processes, supply, development, operation, and maintenance. It also helps with documentation, configuration management, quality assurance, verification, and validation. The standard mainly helps developers by tailoring processes to fit project needs, promoting a systematic and organized approach to software engineering. We were advised to use this system by our course instructor, Mert Bıçakçı, for a smoother and easier development course.

4.2. Risks and Alternatives

ContentGuard aims to make people have a mindful experience on Twitter, track their habits while using the site, and create a report on their time and how they spend it there. The report's main goal is to categorize the tweets the user interacts with or tweets. Several machine learning algorithms will categorize the tweets. However, this approach needs the detailed analysis of a large amount of data to correctly identify the category of a tweet. This also means that the accuracy of the category of a tweet is a significant risk in this project. One tweet’s content can have two categories with equal to near equal similarity. We plan to have quite a large feed for the machine learning algorithms to prevent this close judgment.

Another problem that may occur is the Twitter API usage cost. Currently, the API allows 1500 monthly tweets to be accessed in its free tier. Because of this problem, we cannot train the algorithms that are well enough to just make a judgment, let alone push it onto a level where the algorithms have a high chance of categorizing the tweets with high accuracy. Our solution to this problem is to use Kaggle. Kaggle already has existing data sets that are ready to use, and the data sets are big enough for us to train the algorithms for maximum accuracy.

The following table describes the risks’ likelihood, effect on the project, and our plan B summary.

	Likelihood	Effect on the project	Plan B summary
Data set not big enough	5/10	Program cannot accurately decide on one category for a tweet	Train the ML model with broader dataset

We cannot pay for Twitter API	8/10	We cannot feed enough data for the program to accurately decide on one category for a tweet	Use Kaggel's data sets to train the ML model

Table 1: Risks

4.3. Project Plan

WP#	Work package title	Leader	Members involved
WP1	Web	İlayda Zehra Yılmaz	Everyone
WP2	Extension	Bengisu Buket Karadoğan	Everyone
WP3	Backend	Gülin Çetinus	Everyone
WP4	Model	Burak Öztürk	Everyone
WP5	Testing	Zeynep Derin Dedeler	Everyone

Table 2: Project Package Table

WP 1: *Web*			
Start date: *23.12.2023* **End date:** *29.02.2024*			
Leader:	İlayda Zehra Yılmaz	**Members involved:**	*Everyone*
Objectives: *Web application will be implemented*			

Tasks: ***Task 1.1 Create web page project*****:** *Create the project* ***Task 1.2*** **Implement *functional UI*:** *implementation of designed pages for web user interface* ***Task 1.3 Connect with the backend*****:** *create a connection to the backend* ***Task 1.4 Implement report management*****:** *implement visual components for presenting report contents. synchronize report updates with the backend.* ***Task 1.5 Model integration*****:** *Integrate AI mode to the project*
Deliverables ***D1.1:*** *Web application will be completed* ***D1.2:*** *Report visualization*

Table 3: Package 1: Web

WP 2: *Extension*			
Start date: *23.12.2023* **End date:** 29.02.2024			
Leader:	*Bengisu Buket Kardoğan*	**Members involved:**	*Everyone*
Objectives: *Extension will be implemented*			

Tasks: ***Task 2.1 Create Extension Project*****:** *Create the extension project* ***Task 2.2 Connect with the backend*****:** *create a connection to the backend* ***Task 2.3 Modify Twitter Feed*****:** *Implement operations for modifying the Twitter feed.* ***Task 2.4 Model Integration*****:** *Integrate the AI model and process the Tweet feed.*
Deliverables ***D2.1:*** *Web Extension Project* ***D2.2:*** *Establish interaction of the AI model and Twitter feed.*

Table 4: Package 2: Extension

WP 3: Backend			
Start date: *23.12.2023* **End date:** 15.05.2024			
Leader:	*Gülin Çetinus*	**Members involved:**	*Everyone*
Objectives: *Backend will be implemented*			
Tasks: ***Task 3.1 Create Backend Project*****:** *Create project* ***Task 3.2 Integrate with Frontend*****:** *Create connection with frontend*			

Task 3.3 Model Integration**:** *Integrate model and process Twitter feed*
Deliverables ***D3.1:*** *Backend Application*

Table 5: Package 3: Backend

WP 4: Model			
Start date: 20.12.2023 End date: 08.05.2024			
Leader:	**Burak Öztürk**	**Members involved:**	**Everyone**
Objectives: NLP model will be trained			
Tasks: ***Task 4.1 Create model Project :*** Create project ***Task 4.2 Preprocessing and Collecting Data:*** Collecting and preprocessing Data possibly use existed ml model to generate data ***Task 4.3 Training models:*** Training different models and investigate the optimal model ***Task 4.4 Model Integration:*** Integrate model with backend			
Deliverables ***D4.1:*** *Training and obtaining optimal AI model*			

Table 6: Package 4: Model

WP 5: ***Testing***			
Start date: *04.01.2023* **End date:** *05.06.2024*			
Leader:	**Zeynep Derin Dedeler**	**Members involved:**	**Everyone**

Objectives:
Tasks: ***Task 5.1 Model Testing*****:** *Validating and the testing the AI Model and improving the AI Model* ***Task 5.2 Testing frontend of Web and Extension*****:** *Before connecting to the backend of the project testing the frontend and interactions between web and extension.* ***Task 5.3 Backend API testing*****:** *Backend and Database testing. Request interaction testing.*
Deliverables ***D5.1:*** *Tested and Revised Application*

Table 7: Package 5: Testing

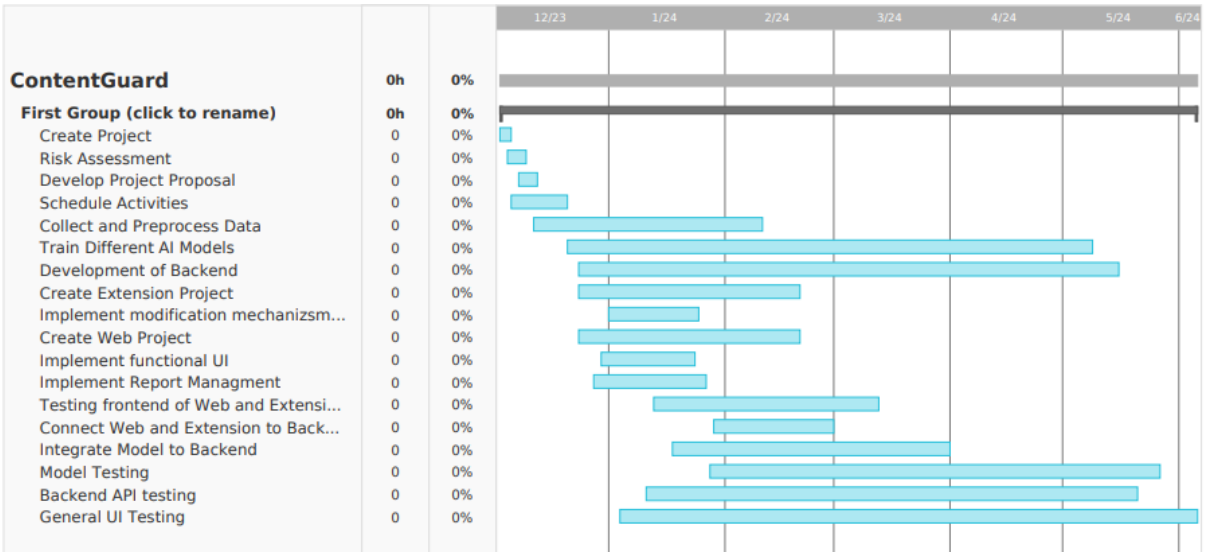

Figure 22: Project Plan Gantt Chart

4.4. Ensuring Proper Teamwork

As Bilkent University students, we put a lot of emphasis and importance on the phases of a project, as we have done many times before. As this is our final project, it is even more important to do our tasks correctly in the right phases.

Because all of us are very busy with many courses, we must plan and divide our tasks correctly. For this, we have discussed some rules. Some of these were about meeting often, mostly every week, either on Zoom or face to face, to discuss our project or our latest report, do some of the crucial starting work together, and then divide the project into equal and fair parts to meet our deadlines. All of the team members agreed on this. Also, when a teammate was busy at some time, we found ways to give more work to the rest of us for that week and give more work to that teammate in the following week or some other report. To do this, it was crucial to maintain good communication and make every team member feel welcome to share their thoughts and not judge, which helped the communication and brainstorming a lot. Also, we asked each other for help or collaborated with each other in sömall groups within the group when one task was beyond us. Group members were sensitive about doing their share of work and not leaving other members too much work. This kind of environment was very good for motivation, especially when we were working together on a task. Even when our schedules were tight, we put effort into meeting each other in small breaks we could find on top of our weekly meetings, and this helped a lot even if we could not always manage this. When necessary, many good members showed leadership skills, being able to take the reins of a situation to make the teamwork work smoothly. We have faced issues but were able to make up for them in the final product by our diligent and collaborative teamwork.

4.5. Ethics and Professional Responsibilities

- While processing the feed of the user, we will process the user's original paragraphs, sentences, and words, which are the product of their creative process. Because this data is sensitive and it is the user's right to share it, we will protect them and not share the data with any other third-party company, person, etc.
- We will store the data in a secure way to be professional in our act of protecting user’s rights.
- When storing the login data for the user, we will store them securely and use hashes to store the passwords to increase privacy and security.
- After a user logs in to Twitter, we are considering giving the user more control by asking them if they want to launch the application or other measures of

control like stopping the application for once choice and stopping for always choice might be given to the user.

- Also, after launching the application to be used by people on the internet, we will make documentation that shows the user to informs the user about these ethical constraints so that they can choose freely to download or not.
- The data will not be stored so that its source can be linked.
- The data will be analyzed and grouped to minimize biases in the subject, and experts may be consulted for determination.
- Our team will follow the Ethics of the National Society of Professional Engineers in this project [1].
- If some features are decided to be paid features or the subscription is added to the Twitter Content Tracker in the future by the development team, it is understood by us that the payments system should be secure, and the team will make their efforts to make it as such.

4.6. Planning for New Knowledge and Learning Strategies

We have decided to do regular research on our main topics like the extensions of this type, AI algorithms, Twitter interface/API, and how these topics can be handled together. For this, we are researching informative videos, similar projects, and useful texts on the internet. Also, our talks with experts like our instructors or other professors in our department will hopefully be helpful in learning how to develop our project. Another important part of this learning process is that there are many other lessons we must also put time into. So we are carefully planning and distributing research subjects between team members and in accordance with everyone's calendars. For these time constraints and the constraint of using a technology we are learning just now, we must be adaptable in the search for new knowledge. Adaptable to the subject’s or technology constraints and the time constraints. To do this, we are asking each other for new information and researching on the internet to find a project/source that has the solution to our questions. We have decided to meet weekly (sometimes 2 times a week) to inform each other about our developments, things we have learned and problems we found possible solutions to. In our research, we are using Github, StackOverflow, YouTube, and many other resources to find our solutions and engage with the experts. Using

trustable resources is very important in learning ethics or the solution to a problem so we are using trustable websites and creditable answers as well as formal documents on our subject. We have many important subjects that we must understand well to be able to implement as we are trying to develop a system that is user-friendly and something that would be usable and understandable to the users.

6. References

[1]NSPE, “NSPE Code of Ethics for Engineers,” *National Society of Professional Engineers*, Jul. 2019. https://www.nspe.org/resources/ethics/code-ethics (accessed Nov. 16, 2023).