Bilkent University
Department of Computer Engineering

Senior Design Project

T2302

NEUTLAN

Detailed Design Report

21801347, Akmuhammet, akmuhammet@ug.bilkent.edu.tr
21803313, Sıla Saraǧlu, sila.saraoglu@ug.bilkent.edu.tr
21703920, Berke Ceran, berke.ceran@ug.bilkent.edu.tr
21801781, Lara Merdol, lara.merdol@ug.bilkent.edu.tr

Can Alkan

Erhan Dolak, Tagmac Topal

06.03.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

1. Introduction

In today's contemporary world, gender equality is one of the outstanding topics. It is an international concept that many international organizations and even countries are trying to raise awareness about. Therefore, there are several actions taken to this issue however there are many aspects of it such as economics, language, and daily life. For example, to close the gender pay gap the globe needs one hundred and thirty-two years [1]. To achieve gender equality in language, we need a Gender-Fair language (GFL). One principle can be obtained to have a gender-fair language: neutralization. Neutralization is changing the words with masculine forms to gender-unmarked forms [2]. For example, today in the formal language and contents, he or she is used to representing a person. Replacing this with they or them is turning the words into an unmarked gender form or instead of using policeman (a masculine form word), police officers can be used to neutralize the word and job in terms of gender. Another significant concept of today's world in terms of gender equality is gender-biased artificial intelligence (AI). Today, we see many examples of gender-biased AI in many technological applications. A classic example is Google Translate. For example, Turkish is gender neutral language in terms of pronouns so if "This person is president, and this person is cooking" is written in Translate, it puts "He's president and she's cooking" [3]. It transforms the sentence in a biased way.

1.1. Purpose of the System

According to the above section, we decided to contribute to making an improvement for gender equality. Our project's name is NEUTLAN which comes from the slogan neutralize the language. In our project, the aim is to neutralize the content of any written text in terms of language by implementing the neutralization principle. In this application, the texts written or uploaded by the users will be checked with the help of Artificial Intelligence and it will be found whether there is any sexist approach in these articles. According to these findings, users will be given warnings and suggestions to ensure that their texts are gender equal. In this way, we will act against gender-biased AI by forming an unbiased one. Our target group for this project is mainly content creators nevertheless; everyone can use it to write more unbiased content.

1.2. Design Goals

1.2.1. User-friendliness

The user interface should be simple and effective in a way that can be easily understood and used by the user. The user interface should not be too complicated because NEUTLAN aims to appeal to literate users of all ages.

1.2.2. Reliability

The project to be done should correctly distinguish sexist patterns, meanings, and words in sentences or texts, and suggest proper corrections. The sexist structures detected should have a high proper accuracy rate.

NEUTLAN’s servers should not be down frequently or for long periods of time because NEUTLAN will become part of users' daily lives.

When an error occurs on the program or the operation that the user wants to perform on the program cannot be realized, the user should be informed adequately.

1.2.3. Performance

The runtime of the application to be made should give feedback to the user as quickly as possible (~20-30s).

1.2.4. Extensibility

The application should be able to be extensible in terms of adding or developing new features according to the feedback from the innovation experts and supervisors.

The extension will be developed primarily for NEUTLAN and then the web application will be developed.

1.2.5. Scalability

NEUTLAN will be developed as a proof of concept application but in the next stage, we aim to reach a wide audience. Therefore, NEUTLAN should be scalable in a way that can reach many users.

1.3. Definitions, acronyms, and abbreviations

- GFL: Gender-fair language (GFL) aims at reducing gender stereotyping and discrimination [6].
- NLP: Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data [7].
- API: The application programming interface can be described as a waiter which ensures one application communicates with another one in transferring stored data or different instructions.
- Chrome Extensions: Chrome extensions are small software programs that beautify and customize people's experience or work while performing various operations within their google chrome pages. They are implemented with web technologies such as HTML, JavaScript, and CSS.

1.4. Overview

In this report and the following sections, we will provide details of our current and developing software architecture system, subsystem, and different fields which are related to our project. With this report, you will have more detailed information about the Neutlan project and more comprehensive information about our system and workload.

2. Current software architecture

In this chapter, we will talk about other systems that are like Neutlan in terms of what they do and how they work. Then, we will explain what makes Neutlan different from these other systems.

2.1. Similar Applications

2.1.1 Grammarly

Grammarly is a web-based AI-powered writing assistant that helps users improve their writing by detecting and correcting grammar, spelling, punctuation, and style errors. It offers a wide range of features and functionalities such as grammar checking, spelling checking, vocabulary enhancement, and so on. As our application, Grammarly also can be used as its web application and also as an extension. Furthermore, it has its own mobile and desktop application. Grammarly uses a combination of natural language processing (NLP), machine learning, and artificial intelligence (AI) algorithms to analyze text and provide feedback on grammar, spelling, punctuation, and style. In our system design, we have taken the example of this application system design considering it is widely used and practical. As we discuss in other reports we have also utilized the NLP model for sexist content detection [4].

2.1.2 Google Translate

Google Translate is a web-based machine translation tool that enables users to translate text, speech, images, and entire websites between different languages. It offers a variety of functionalities and features that cater to different translation needs and preferences, including; text and speech translation and so on. The design of Google Translate is based on a complex and evolving set of machine learning algorithms that analyze and interpret input text, translate it into a target language, and then generate output text that is grammatically correct and semantically meaningful. The system is built on a massive database of text and language models that are continually refined and updated based on feedback from users and language experts. Gender-neutral language is an important consideration in the design of Google Translate, as it is designed to be inclusive and respectful of all users, regardless of their gender identity or expression. To achieve this, the system employs a variety of techniques, such as

using gender-neutral pronouns, avoiding gendered language where possible, and providing alternative translations for words and phrases that have gendered connotations in some languages. As Google Translate we have also carefully chosen the dataset that we have used for the recommendation model to be able to recommend and purify users text [5].

2.2. Neutlan

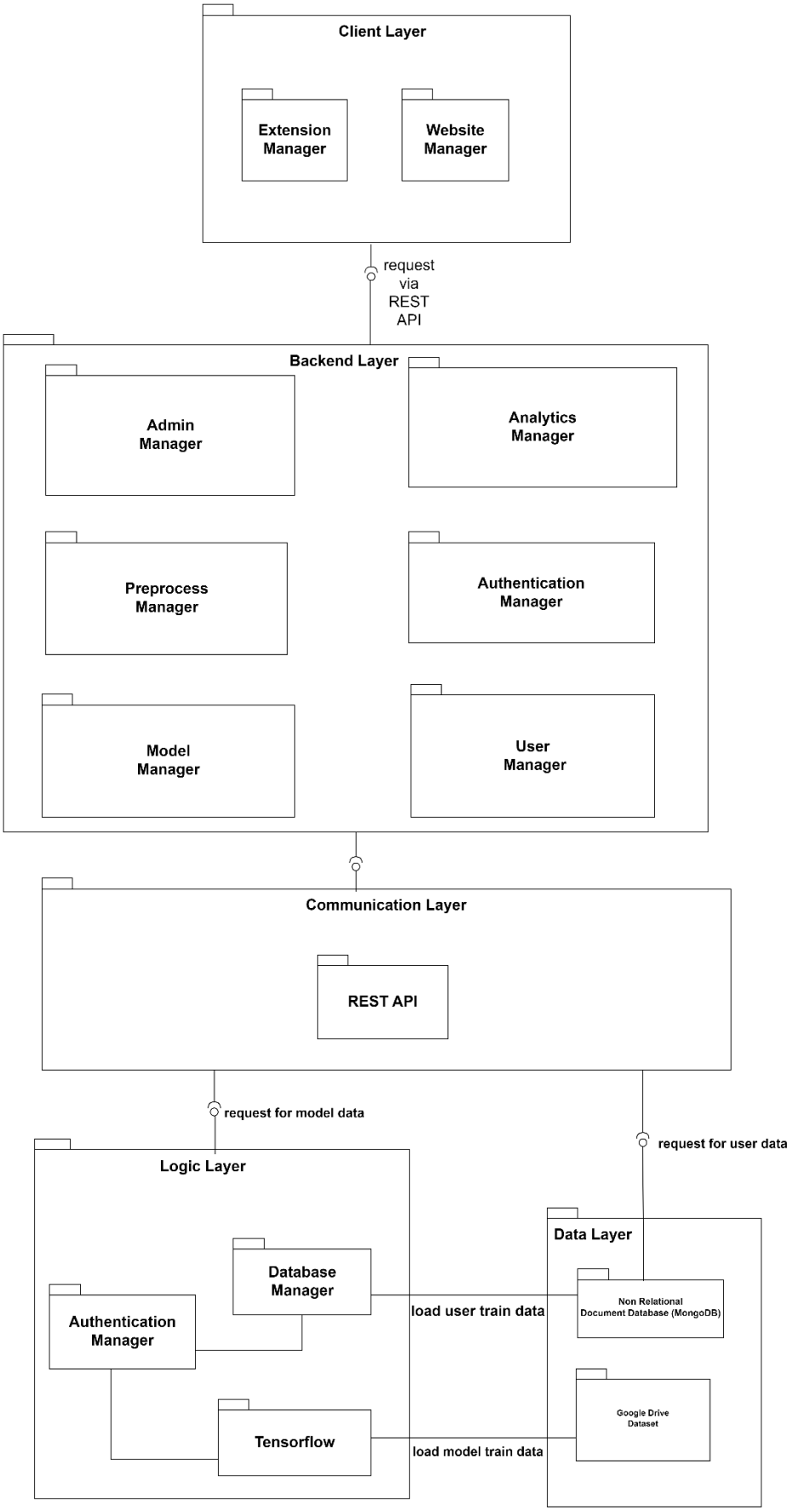
In our analysis report, we highlight that Neutlan is a new and unique project. While research has been conducted on gender-biased content datasets, there are currently no practical applications that can assist people in their daily lives. We also mention some existing applications that share similarities with our project in terms of system design and functionality, and we drew inspiration from these projects during the development of our application. However, it's important to note that Neutlan is distinct from these other applications in its objective to neutralize language, rather than merely checking or translating text. Moreover, our application's extension tool can be used in conjunction with those two applications, making them complementary rather than competitive.

3. Proposed software architecture

3.1. Overview

In this part, the software architecture we designed is explained in a subsystem decomposition diagram. The diagram contains the main layers of our system and the managers in them indicate their services inside the layers. Then, hardware/software mapping of the system will be given. Afterward, a persistent data management section is given to demonstrate Neutlan’s database system. The access control and security part indicates details about our system’s security measurements.

3.2. Subsystem decomposition



Neutlan is based on client-server architecture. It is composed of layers: client, backend, communication, logic, and data layer. Most of the system's actions or calculations will be based on client requests which is why we create our system on the basis of client-server architecture. The client layer is the UI interface which is responsible for sending user requests to the backend for a response. The backend layer is responsible for various actions such as including user requests and communicating with the model logic. The logic layer is represented because of the server logic. It is responsible for handling the logic of the model. Finally, the data layer is represented for our database and datasets. Between our layers, we have the communication layer which consists of API and is responsible for secure and fast communication.

3.3. Hardware/software mapping

Neutlan is a software application that includes no hardware except a mobile or computer. It serves as a website and an extension so it will be available on a mobile or a computer. The extension service works only in Google Chrome however the website is available for all web browsers.

The client part of the software is built via React, while the backend part of the system is built with Django. Their communication is provided by the REST API. For model logic, Tensorflow and for developing it Google Colab is used. For keeping the user data, a non-relational database MongoDB is used and the datasets for the model are kept in a Google Drive. All systems reside in a server that is provided by Amazon AWS Lightsail service.

3.4. Persistent data management

In Neutlan, the objects such as User, File, and Analytics need to be stored in order to provide functionalities as well as enhance user experience. Therefore, we need the database to meet our requirements. The place where those data will be stored should not only be optimized for queries but also secure since the files that are being processed might possess sensitive information of the user. Additionally, the privacy of the user's information such as email and password needs to be taken into account seriously. Hence, we decided to use MongoDB which ensures speed and security. The File objects are also stored here as MongoDB allows storage of files. MongoDB has a library called Pymongo on Python which contains an interface for

queries. We decided to use that library to access data because our backend is implemented with Django. The maintenance and persistence are already done by MongoDB but additional functions will be implemented to make sure that the data is consistent. These functions will only be executed by the admin.

3.5. Access control and security

There will be 2 main users of our application; regular users and admins. Regular users are allowed to use the application and the scope of the data access will be limited to their files and data. Each user will be able to access, modify and delete their data including name, email, password, and files. There will be no interaction between users therefore, users will not be able to access or modify other people's data. This limitation is set to meet the privacy of the user and also, our application does not require interaction between users. Users will be offered an option whether they want to share their data for training purposes or not. The data which will be provided to admins will be anonymous and not public. After training our model with new data, old ones will be deleted.

On the other hand, admins will have full access to all users' data and the authority to modify if it is required. Such extensive power is provided to admins since it is essential to maintain a database in terms of persistence and consistency. However, admins will not access or modify users' data unless there is no necessity. The shared data for boosting the accuracy of the NLP model will be preprocessed first to provide anonymity to the users and used for training purposes.

Access to the data will be only through the server including admin interventions. Only anonymized data will be transferred to Google Collab with the existing NLP model to achieve transfer learning and data will be deleted after achieving that goal.

Lastly, all users will be able to access their data through tokens which will be provided in the login step. That token will be stored in Chrome's cache and used when it is required. Each token will be unique and assigned to the user at each sign-in and deleted with the stage of log out.

4. Subsystem services

In this section, the details of the services inside the layers which were explained above are given. In all of the layers, there are managers which are responsible for the several services inside the particular layer.

4.1. Client Layer

Client layer is the one which accounts for the UI interface and its actions. Neutlan provides services both for Chrome extension and website thus there are two managers for both.

4.1.1. Extension Manager

Extension manager is responsible for the actions in the extension. It is developed with the help of JavaScript and it is available as a Chrome extension. This manager indicates the content user is written in the browser and sends it for evaluation.

4.1.2. Website Manager

Website manager is responsible for the actions on the website. It is developed with React framework and is available in any browser. This manager includes functionalities like writing on an editor, file uploading, and profile functionalities

4.2. Backend Layer

Backend layer of the project is built with the Django framework in Python language and is responsible for getting the client requests from API and performing actions accordingly. Backend layer also uses the Pymongo module to communicate with the MongoDB database and for its functions. This module also provides communication with the database and Django via REST API.

4.2.1. Admin Layer

Admin layer is for the representation of the Django admin in the system. Since the project is developed in Django, we have an admin from Django which comes as default in the

projects. Via this admin, we can get help with token management. Furthermore, from this admin manager, we have provided functions in which we can update the database with the information filled in the Django admin or we can delete all information both in the Django admin and database.

4.2.2. Analytics Manager

Analytics manager is for the calculations of the project analysis. This manager calculates several analytics of the application such as the number of total female and male users in the project, the total number of edited or uploaded documents, and so on.

4.2.3. Authentication Manager

Authentication manager accounts for the user authentication functionalities such as sign in, sign up, and forget password. We use the Pymongo module and Rest framework for the database and its connection. For email confirmation, in this manager, smtplib module and JSON Web Tokens are used.

4.2.4. User Manager

User manager is responsible for the user functionalities such as editing profile information, deleting their account, uploading and retrieving a file, and so on. In this manager, in addition to the other modules we use GridFS for uploading files to MongoDB which are bigger than 16 MB. In this way, the system could store all of the text fields which are submitted.

4.2.5. Model Manager

This manager is responsible for sending the text which the user gives to the application to the evaluation for biased content. Meanwhile, before sending it to evaluation it passes the content to preprocess manager and tokenizer for data to be cleaned and organized for the prediction model. Furthermore, there is an additional manager for paraphrasing models. This manager also utilizes preprocess manager.

4.2.6. Preprocess Manager

This manager is responsible for processing data (content the user gives) before the model evaluation. This manager organizes content that will be the input for the model evaluation. It is built with the help of the Tensorflow module.

4.3. Communication Layer

Communication layer is the representation of the layer between the following layers: client and backend, backend and logic, and backend and data layers. All of the communication between the layers is done with the help of the REST framework of REST API.

4.4. Logic Layer

Logic layer is the representation of the system's server which hosts the Tensorflow and database manager.

4.4.1. Authentication Manager

This layer is responsible for the authentication on the server. It is independent from the user functionality part of the project. It is used for access to servers with various other functionalities for developers.

4.4.2. Database Manager

This layer is responsible for database management within the server.

4.4.3. Tensorflow

This layer accounts for the model prediction and evaluation functionalities. This is kept within the server so the project can access it from anywhere contrary to working on local servers.

4.5. Data Layer

Data layer is the representation of our project's database where we keep our datasets for our model and information about our users.

4.5.1. Non-relational Document Database (MongoDB)

This database keeps the user information of our project. The data is kept in the MongoDB Atlas server which is non-local thus it can be accessed from anywhere. The GridFS feature in MongoDB is also being used for text file type documentation.

4.5.2. Google Drive Dataset

This Google Drive repository keeps the datasets which are crucial for our model. Based on these datasets, our model determines biased content verification.

5. Test Cases

We provided 52 test cases for all Extension, Web application, Backend, Model and Database below. The cases related to extension have the following pattern: *field-category-id*. The encodings of field and category as follows:

Field	
Backend	bac
Extension	ext
Web	web
Model	mod
Database	db

Category	
Performance	per
Functionality	fun
UI	ui
Security	sec
Stress	str

All warnings that are mentioned below means that the system will not execute the function since.

Test ID	ext-ui-001	Category	UI	Severity	Critical
Objective	Test case for detection of biased sentences being written on other websites.				
Steps	1. Open Google translate. 2. Write “Men are stronger than women.”				
Expected	The sentence “Men are stronger than women.” is biased and should be underlined with extension.				
Date-Result	12/2022 - pass				

Test ID	ext-fun-001	Category	Functionality	Severity	Critical
Objective	Test case for suggestion to biased sentences being written on other websites.				
Steps	1. Open Google translate. 2. Write “Men are stronger than women.”				
Expected	The sentence “Men are stronger than women.” is biased and it will be underlined with extension. Then a suggestion for this sentence will appear for change.				
Date-Result	04/2023 - ?				

Test ID	web-ui-001	Category	UI	Severity	Critical
Objective	Test case for detection of biased sentences being written on a website.				
Steps	1. Open editor of our website. 2. Write “Men are stronger than women.”				
Expected	The sentence “Men are stronger than women.” is biased and should be underlined.				
Date-Result	04/2023 - ?				

Test ID	web-ui-002	Category	UI	Severity	Minor
Objective	Test case for welcome page slider of website.				
Steps	1. Open neutlan website. 2. Wait 6 seconds.				
Expected	Images of welcome page should be change every 3 seconds.				

Date-Result	02/2023 - pass

Test ID	web-ui-003	Category	UI	Severity	Minor
Objective	Test case for hiding the password in the input section in login page of website.				
Steps	1. Open neutlan login web page. 2. Write “Berke123” password to the password field.				
Expected	The password “Berke123” should be hidden.				
Date-Result	04/2023 - ?				

Test ID	web-ui-004	Category	UI	Severity	Minor
Objective	Test case for loading spin in the editor page of website.				
Steps	1. Open editor page. 2. Write “Men are stronger than women.”.				
Expected	Loading spin should be appear until backend send response of biased sentence.				
Date-Result	04/2023 - ?				

Test ID	web-ui-005	Category	UI	Severity	Major
Objective	Test case for website responsivenes.				
Steps	1. Open neutlan website. 2. Open developer mode. 3. Change dimension of webpage.				
Expected	According to dimension, website should be responsive.				
Date-Result	04/2023 - ?				

Test ID	web-fun-001	Category	Functional	Severity	Minor
Objective	Test case for valid email in input-field in sign up page of website.				
Steps	1. Email field will be filled with "akmuhammet99@hotmail.com" and "akmuhammet99.gmail.com" (as 2 scenarios). 2. Enter Test1234. for both passwords fields. 3. Click the Sign Up button.				
Expected	The website should allow "akmuhammet99@gmail.com" and give a warning for "akmuhammet99.gmail.com" as incorrect email format.				
Date-Result	12/2022 - pass				

Test ID	web-fun-002	Category	Functional	Severity	Minor
Objective	Test case for password input field in sign up page of website. Passwords should contain at least one lower case, upper case, and digit. Also, the password should be at least 8 characters.				
Steps	1. "Berke123", "berke123", "BERKE123" and "Berk123" will be entered into password fields and "akmuhammet99@hotmail.com" will be entered as an email input (as 4 scenarios). 2. Click the Sign Up button.				
Expected	Only "Berke123" will be accepted as a correct password input. For the rest of the cases, a warning message will appear accordingly.				
Date-Result	12/2022 - pass				

Test ID	web-fun-003	Category	Functional	Severity	Minor
Objective	Test case for password match in input field in sign up page of website.				
Steps	1. Enter "akmuhammet99@gmail.com" for the email and "Berke123" for the password field. 2. Enter "Berke123" and "Berke1234" for password again field as a second scenario (as second scenario). 3. Click the Sign In button.				
Expected	For the password again input of "Berke123" website should accept and for "Berke1234" it will give warning.				
Date-Result	12/2022 - pass				

Test ID	web-fun-004	Category	Functional	Severity	Critical
Objective	Test case for sign up where the user already exists in the sign up page of the website.				
Steps	1. Fill the email and passwords fields with "berke.ceran@gmail.com", "Berke123" and "Berke123", respectively (as 2 scenarios). 2. Click the Sign In button.				
Expected	System will return a "User already exists" warning.				
Date-Result	12/2022 - pass				

Test ID	web-fun-005	Category	Functional	Severity	Minor
Objective	Test case for sign in button functionality in sign up page of website.				
Steps	1. Click the Sign In button on the sign up page.				
Expected	System will navigate to the Sign In page.				
Date-Result	12/2022 - pass				

Test ID	web-fun-006	Category	Functional	Severity	Critical
Objective	Test case for login functionality in sign in page of website.				
Steps	1. Enter "berke.ceran@gmail.com" and "pass1234" for email and password on the sign in page. 2. Click the Sign In button.				
Expected	System will navigate to the main page as the credentials are correct.				
Date-Result	12/2022 - pass				

Test ID	web-fun-007	Category	Functional	Severity	Minor
Objective	Test case for Remember Me functionality in sign in page of website.				
Steps	1. Enter "berke.ceran@gmail.com" and "pass1234" for email and password on the sign in page. 2. Click the Sign In button.				
Expected	If there is a successful login, user's email address and password should remain saved.				
Date-Result	12/2022 - pass				

Test ID	web-fun-008	Category	Functional	Severity	Minor
Objective	Test case for the forget password button functionality in the sign in page of the website.				
Steps	1. Click Forget Password button in sign in page				
Expected	System will navigate to the forget password page.				
Date-Result	12/2022 - pass				

Test ID	web-fun-009	Category	Functional	Severity	Minor
Objective	Test case for navigating to the sign up page in sign in page of website.				
Steps	1. Click the Sign Up button in sign in page				
Expected	System will navigate to the sign up page.				
Date-Result	12/2022 - pass				

Test ID	web-fun-010	Category	Functional	Severity	Critical
Objective	Test case for sending reset password email for the forget password functionality.				
Steps	1. Enter “berke.ceran@gmail.com” in the forget password page in the email input field. 2. Click the Reset Password button.				
Expected	The email should be sent to “berke.ceran@gmail.com” that contains instructions for resetting the password.				
Date-Result	12/2022 - pass				

Test ID	web-fun-011	Category	Functional	Severity	Critical
Objective	Test case for reset password functionality with instructions being provided in email.				
Steps	1. Follow steps in test case web-fun-010 (if not done) 2. Click the Reset Password button being provided in email. 3. Enter “newpass123” and “newpass123” for passwords in the navigated site from mail. 4. Click the Reset Password button.				
Expected	System will show “Password Changed Succesfully.” message.				

Date-Result	12/2022 - pass

Test ID	web-fun-012	Category	Functional	Severity	Critical
Objective	Test case for login with new password in sign in page of website				
Steps	1. Follow steps in test case web-fun-011 (if not done) 2. Enter “berke.ceran@gmail.com” and “newpass123” as an email and password in the sign in page.				
Expected	System will navigate to the main page as credentials will be correct.				
Date-Result	12/2022 - pass				

Test ID	web-fun-013	Category	Functional	Severity	Critical
Objective	Test case for Reset Password after 15 minutes by following instructions in email being sent.				
Steps	1. Follow steps in test case web-fun-010. 2. Wait for 16 minutes. 3. Click the Reset Password button provided in mail.				
Expected	System will navigate to the web page were it will show “Expired Link” warning.				
Date-Result	12/2022 - pass				

Test ID	web-fun-0014	Category	Functional	Severity	Critical
Objective	Test case for opening the home page correctly.				
Steps	1. Sign In with “berke.ceran@gmail.com” and “pass1234”.				
Expected	System will navigate to the home page that belongs to “berke.ceran@gmail.com” profile.				
Date-Result	02/2023 - pass				

Test ID	web-fun-015	Category	Functional	Severity	Major
Objective	Test Case for uploading text file in home page of website				
Steps	1. Click the upload file button on the home page.				

Expected	New window that will allow only the selection of .txt files will appear for uploading.
Date-Result	02/2023 - pass

Test ID	web-fun-016	Category	Functional	Severity	Critical
Objective	Opening document of a user in home page of website				
Steps	1. Click the test.txt file on the home page.				
Expected	System will navigate to the editor page with the contents of test.txt files.				
Date-Result	03/2023 - pass				

Test ID	web-fun-017	Category	Functionality	Severity	Critical
Objective	Test case for suggestion to biased sentences being written on website.				
Steps	1. Open neutlan editor. 2. Write “Men are stronger than women.”				
Expected	The sentence “Men are stronger than women.” is biased and it will be underlined. Then a suggestion for this sentence will appear for change.				
Date-Result	04/2023 - ?				

Test ID	mod-fun-001	Category	Functional	Severity	Critical
Objective	This test case is to verify that the Neutlan NLP model for biased detection is able to detect positive bias in language.				
Steps	1. Execute the NLP model on the sentence "Men are naturally better leaders than women.". 2. Verify that the NLP model detects the positive bias in the sentence and flags it as biased language. 3. If the NLP model detects a positive bias, mark the test case as passed. 4. If the NLP model does not detect the positive bias, investigate the issue and retest after fixing the issue.				

Expected	The NLP model should correctly detect bias in the input sentence and flag it as biased language.
Date-Result	12/2022 - pass

Test ID	mod-fun-002	Category	Functional	Severity	Critical
Objective	This test case is to verify that the Neutlan NLP model for biased detection is able to detect negative bias in the inputted sentence.				
Steps	1. Input a sentence that contains negative bias, such as "Women are not as good at math as men." 2. Execute the NLP model on the sentence. 3. Verify that the NLP model detects the negative bias in the sentence and flags it as biased language. 4. If the NLP model detects the negative bias in the sentence, mark the test case as passed. 5. If the NLP model does not detect the negative bias, investigate the issue and retest after fixing the issue.				
Expected	The NLP model should correctly detect bias in the input sentence and flag it as biased language.				
Date-Result	12/2022 - pass				

Test ID	mod-fun-003	Category	Functional	Severity	Critical
Objective	This test case is to verify that the Neutlan NLP model for biased detection does not flag neutral language as biased language.				
Steps	1. Execute the NLP model on the sentence "People are good." 2. Verify that the NLP model does not flag the sentence as biased language. 3. If the NLP model does not flag the sentence as biased language, mark the test case as passed. 4. If the NLP model incorrectly flags the sentence as biased language, investigate the issue and retest after fixing the issue.				
Expected	The NLP model should correctly identify the input sentence as neutral language and not flag it as biased language.				

Date-Result	12/2022 - pass

Test ID	mod-fun-004	Category	Functional	Severity	Major
Objective	This test case is to verify that the Neutlan NLP model is able to function correctly in the Docker environment.				
Steps	1. Deploy the TensorFlow Serving container on Docker. 2. Start the container and confirm that it is running correctly. 3. Connect to the TensorFlow Serving container using the API. 4. Execute the NLP model using a test sentence. 5. Verify that the NLP model returns the expected output for the test sentence. 6. If the NLP model returns the expected output, mark the test case as passed. 7. If the NLP model does not return the expected output, investigate the issue and retest after fixing the issue.				
Expected	The NLP model should be able to function correctly in the Docker environment and return the expected output for the input sentence.				
Date-Result	01/2023 - pass				

Test ID	mod-fun-005	Category	Functional	Severity	Major
Objective	To test the accuracy of the biased sentence detection prediction function in REST API.				
Steps	1. Send a POST request to the REST API prediction function with a biased sentence containing biased language within the server. Example: {"sentence": "Women are not fit for leadership roles."} 2. Verify that the API returns a response with a boolean value of "True". 3. Send a POST request to the REST API prediction function with an unbiased sentence that doesn't contain any discriminatory language. Example: {"sentence": "The sky is blue."} 4. Verify that the API returns a response with a boolean value of "False".				
Expected	Step 2 should pass and the API should return "True" for biased sentences. Step 4 should pass and the API should return "False" for unbiased sentences.				

Date-Result	01/2023 - pass

Test ID	mod-sec-006	Category	Security	Severity	Major
Objective	To test the authentication requirements of the Biased Sentence Detection API.				
Steps	1. Attempt to send a POST request to the Biased Sentence Detection API endpoint without including an authentication token. 2. Provide a sample text input to the API. 3. Verify that the API response returns an error message stating that authentication is required to use the prediction method. 4. Record the test result as passed if it returns an error message stating that authentication is required.				
Expected	It should return an error message stating that authentication is required				
Date-Result	01/2023 - pass				

Test ID	mod-sec-007	Category	Security	Severity	Major
Objective	To test the valid authentication requirements of the Biased Sentence Detection API.				
Steps	1. Attempt to send a POST request to the Biased Sentence Detection API endpoint with an invalid authentication token. 2. Provide a sample text input to the API. 3. Verify that the API response returns an error message stating that the provided authentication token is invalid or expired. 4. Record the test result as passed if it returns an error message stating that the provided authentication token is invalid .				
Expected	It should return an error message stating that the authentication token is invalid or expired.				
Date-Result	01/2023 - pass				

Test ID	mod-sec-008	Category	Security	Severity	Major

Objective	To test the valid authentication requirements of the Biased Sentence Detection API.
Steps	1. Attempt to send a POST request to the Biased Sentence Detection API endpoint with a valid authentication token. 2. Provide a sample text input to the API. 3. Verify that the API response returns a prediction for the provided text input. 4. Record the test result as passed if it returns a valid prediction.
Expected	It should return a valid prediction.
Date-Result	01/2023 - pass

Test ID	mod-per-009	Category	Performance	Severity	Major
Objective	To test the performance of the Neutral NLP model for biased language detection.				
Steps	1. Send a POST request to the Neutlan NLP model API endpoint with the test sentence as input. 2. Record the start time of the API response. 3. Verify that the API response correctly identifies positive bias in the input sentence and flags it as biased language. 4. Record the end time of the API response. 5. Calculate the response time by subtracting the start time from the end time. 6. If the response time is under 30 second, mark the test case as passed. 7. If the API response does not correctly identify positive bias in the input sentence, record the test result as a failure and investigate the issue.				
Expected	The API response time should be under 30 second.				
Date-Result	01/2023 - pass				

Test ID	mod-fun-010	Category	Functional	Severity	Critical
Objective	To test the accuracy of the Paraphrasing model in removing positive bias from a biased sentence.				

Steps	1. Input a sentence that contains positive bias, such as "Men are naturally better leaders than women." 2. Execute the NLP paraphrasing model of the sentence. 3. Verify that the model response returns a recommended paraphrased sentence that is not biased such as “Men and women are both better leaders”. 4. Input the recommended sentence into the Neutlan sexist content detection NLP model. 5. If the recommended paraphrased sentence is not biased, mark the test case as passed. 6. If the recommended paraphrased sentence still contains biased language, record the test result as a failure and investigate the issue.
Expected	The Paraphrasing Model should recommend a paraphrased sentence that is not biased. Note: This test case should be repeated with multiple test sentences to ensure consistent and accurate results. Additionally, a human reviewer should verify that the recommended paraphrased sentence is not biased.
Date-Result	02/2023 - pass

Test ID	mod-fun-011	Category	Functional	Severity	Critical
Objective	This test case is to verify that the Neutlan NLP model for biased detection is able to detect negative bias in the inputted sentence.				
Steps	1. Input a sentence that contains negative bias, such as "Women are not as good at math as men." 2. Execute the NLP paraphrasing model of the sentence. 3. Verify that the model response returns a recommended paraphrased sentence that is not biased such as “Men and women are equally good at math.”. 4. Input the recommended sentence into the Neutlan sexist content detection NLP model. 5. If the recommended paraphrased sentence is not biased, mark the test case as passed. 6. If the recommended paraphrased sentence still contains biased language, record the test result as a failure and investigate the issue.				

Expected	The Paraphrasing Model should recommend a paraphrased sentence that is not biased. Note: This test case should be repeated with multiple test sentences to ensure consistent and accurate results. Additionally, a human reviewer should verify that the recommended paraphrased sentence is not biased.
Date-Result	02/2023 - pass

Test ID	mod-fun-012	Category	Functional	Severity	Critical
Objective	To test the accuracy of the Paraphrasing Model API.				
Steps	1. Send a POST request to the Sexist Content Detection Model API endpoint with the test sentence as input. 2. Verify that the API response correctly identifies the sentence as containing sexist or biased language. 3. If the API response does not identify the sentence as containing sexist or biased language, record the test result as a failure and investigate the issue. 4. Send a POST request to the Paraphrasing Model API endpoint with the test sentence as input. 5. Verify that the API response returns a recommended paraphrased sentence that is not biased. 6. If the recommended paraphrased sentence is not biased, mark the test case as passed. 7. If the recommended paraphrased sentence still contains biased language, record the test result as a failure and investigate the issue.				
Expected	The Sexist Content Detection Model should correctly identify the test sentence as containing biased language. In that case the Paraphrasing Model should recommend a paraphrased sentence that is not biased.				
Date-Result	03/2023 - ?				

Test ID	mod-sec-013	Category	Security	Severity	Major
Objective	To test the authentication requirements of the Paraphrasing Model API.				

Steps	1. Attempt to send a POST request to the Paraphrasing Model API endpoint without including an authentication token. 2. Provide biased input sentences to the API. 3. Verify that the API response returns an error message stating that authentication is required to use the prediction method. 4. Record the test result as passed if it returns an error message stating that authentication is required.
Expected	It should return an error message stating that authentication is required
Date-Result	03/2023 - ?

Test ID	mod-sec-014	Category	Security	Severity	Major
Objective	To test the valid authentication requirements of the Paraphrasing Model API.				
Steps	1. Attempt to send a POST request to the Paraphrasing Model API endpoint with an invalid authentication token. 2. Provide biased input sentences to the API. 3. Verify that the API response returns an error message stating that the provided authentication token is invalid or expired. 4. Record the test result as passed if it returns an error message stating that the provided authentication token is invalid .				
Expected	It should return an error message stating that the authentication token is invalid or expired.				
Date-Result	03/2023 - ?				

Test ID	mod-sec-015	Category	Security	Severity	Major
Objective	To test the valid authentication requirements of the Paraphrasing Model API.				
Steps	1. Attempt to send a POST request to the Paraphrasing Model API endpoint endpoint with a valid authentication token. 2. Provide biased input sentences to the API. 3. Verify that the API response returns a recommended paraphrased sentence that is not biased.				

	4. Record the test result as passed if it returns a valid prediction.
Expected	It should return a valid nonbiased recommendation.
Date-Result	01/2023 - pass

Test ID	mod-per-016	Category	Performance	Severity	Major
Objective	To test the performance of the Neutral Paraphrasing model.				
Steps	1. Send a POST request to the Neutlan Paraphrasing model API endpoint with the biased sentence as input. 2. Record the start time of the API response. 3. Verify that the API response is a valid unbiased recommendation for the inputted sentence. 4. Record the end time of the API response. 5. Calculate the response time by subtracting the start time from the end time. 6. If the response time is under 30 seconds, mark the test case as passed. 7. If the API response does not recommend a non biased paraphrase, record the test result as a failure and investigate the issue.				
Expected	The API response time should be under 30 seconds.				
Date-Result	04/2023 -?				

Test ID	db-fun-001	Category	Functional	Severity	Major
Objective	To test that user file is uploaded into database correctly				
Steps	1. Upload a file from website for a specific user 2. Check that in user collection for particular user, the filename with the file's id is added from MongoDB Atlas 3. Check that this id has a correspondence in fs.chunks and fs.files which involves the details of the file 4. Format the binary data in database to file form from an online tool 5. Check that found content is the same with the uploaded file's content				

	6. Check that uploaded time in the user collection is the same as current date
Expected	1. Converted content from binary data should be the same with the uploaded content 2. Filename in the user collection should match the filename in fs.files collection which both has the same id 3. Upload time in the user collection for the particular file should be same as the current's date
Date-Result	04/2023 -?

Test ID	db-fun-002	Category	Functional	Severity	Major
Objective	To test that edit file correctly works in database				
Steps	1. Edit a file from the website with a filename already exists in the system 2. Check that in user collection for particular file, last edited time is added as the current data and file id is renewed 3. Check that in fs.files and fs.chunks, file the renewed id exists with the new content 4. Check that old file with the old id is deleted in fs.chunks and fs.files collection 5. Format the new formed binary data to file and compare it with the edited content				
Expected	1. The last edited time for particular file is added same as the current date 2. The upload time for the edited file is the same as the old's file upload time 3. File information with the old id must be deleted from fs.chunks and fs.files collections 4. The new file content must be the same as the formatted binary data				
Date-Result	04/2023 -?				

Test ID	db-fun-003	Category	Functional	Severity	Major
Objective	To test the delete file functionally correctly works in the database				

Steps	1. Select a file from a user's account and check the file id 2. Delete the file from particular user's account from website application 3. Check the deleted file information in user collection for the particular user and file is deleted 4. For particular file id, check the fs.chunks and fs.files collection information are deleted
Expected	1. In the user collection, for particular user and file, the information for deleted file does not exist anymore 2. In the fs.chunks and fs.files collections, the file information is deleted
Date-Result	04/2023 -?

Test ID	db-fun-004	Category	Functional	Severity	Major
Objective	To test that forget password functionality correctly works in database				
Steps	1. Perform a forget password request for a particular user from the website application 2. Check that the request with the time that is sent is inserted into the reset password request collection in database				
Expected	The request must be visible in the reset password collection with the request sent time				
Date-Result	04/2023 -?				

Test ID	db-fun-005	Category	Functional	Severity	Major
Objective	To test the change password functionality from the forget password functionality				
Steps	1. Perform a forget password request for a particular user 2. The click the link from mail address to change the password 3. Check the forgotten password from the database in the specific user's document 4. Perform the change password functionality from website				

	5. Check that the new password given is inserted in the user collection as password for particular user 6. Check that old password is inserted into the last password's array in the specific user's document in the collection
Expected	1. The old password must be visible in the last password's array in the specific user's document. 2. The oldest password in the last password's array must be dropped (no longer visible) 3. New password is visible as the password property in the specific user's document in the user collection
Date-Result	04/2023 -?

Test ID	db-fun-006	Category	Functional	Severity	Major
Objective	To test the delete account functionality correctly works in the database				
Steps	1. Select a user from database in the user collection and denote its' information 2. Delete user's account from the website application 3. Check the particular user in the user collection 4. Check the file information for that user in fs.files and fs.chunks collections 5. Check the forget password requests for the particular user in the reset password collection				
Expected	1. The specific user information in the user collection must be no longer visible 2. The uploaded files for the specific user must be deleted from fs.chunks and fs.files collections 3. The reset password requests for the specific user must be no longer visible				
Date-Result	04/2023 -?				

Test ID	db-fun-007	Category	Functional	Severity	Major
Objective	To test the sign up functionality works correctly in the database				

Steps	1. Perform a sign up functionality from the website application 2. Check the user information in the user collection in the database
Expected	1. The same email is inserted in the user document in the user collection 2. The same password is inserted in the specific document 3. The last password array is inserted and empty in the document 4. The files array is inserted and empty in the document 5. The settings object is inserted into the document with default inputs as dark_mode_activated: False, extension-activated: False
Date-Result	04/2023 -?

Test ID	db-str-001	Category	Non-Functional Stress	Severity	Moderate
Objective	To test the function handling capacity of database				
Steps	1. Open many various functions in the website application in different pages of the browser such as opening a page for sign up, opening a page for sign, for upload file and so on 2. Perform each function consecutively 3. Check the each performance's outcome in database				
Expected	At least 15 functionality occurring at the same need to have right outcome in database				
Date-Result	04/2023 -?				

Test ID	db-per-001	Category	Performance	Severity	Moderate
Objective	To test the performance of the upload file functionality in the database				
Steps	1. Perform an upload file functionality for a specific file which has an average size of 0,5 KB 2. Check the file information in fs.files and fs.chunks collections 3. Check the file information in specific user in the user collection				

Expected	The file of an average of 0,5 KB size must be inserted into the database
Date-Result	04/2023 -?

Test ID	bac-fun-001	Category	Functional	Severity	Major
Objective	To test the sign in functionality in REST API				
Steps	1. Prepare a POST request to the REST API with the link: https://neutlan.com/api/auth/sign_in in Postman 2. Add the parameters as email and password in the request for a user which exists in the system 3. Denote the API response as a successful sign in 4. Prepare the post request again 5. Add the parameters as email and password for a non-existent user 6. Denote the API response as an unsuccessful sign in				
Expected	1. In step 3, for a successful sign in, a token must be returned in the HTTP response which holds the value 200 2. In step 6, for an unsuccessful sign in, an error message must be returned in the HTTP response which holds the value 400				
Date-Result	04/2023 -?				

Test ID	bac-per-001	Category	Performance	Severity	Major
Objective	To test the sign in functionality's performance in REST API				
Steps	1. Perform a sign in functionality REST API request in Postman for successful and unsuccessful sign in as in the Test ID: back-fun-001 2. Denote the response times for successful and unsuccessful sign in REST API requests				

Expected	The responses for the successful and unsuccessful sign in REST API requests must be visible in Postman within 1 second.
Date-Result	04/2023 -?

Test ID	bac-fun-002	Category	Functional	Severity	Major
Objective	To test the get file content functionality in REST API				
Steps	1. Select a current user in database with one of their files 2. Denote the current user's email, password and filename information for the specific file 3. Send a sign in request as in the TEST ID: back-fun-001 for successful login with the denoted email and password 4. Denote the token from the response 5. Prepare a GET request with the following link: https://neutlan.com/api/user/get_file_content in the Postman 6. Add the parameter denoted filename as filename in the body of the request 7. Add the token as the authentication token in the request 8. Send the get file content request in Postman 9. Denote the response				
Expected	The file_content in the response should be same as the file content in the database and the HTTP response should have the value 200 for successful				
Date-Result	04/2023 -?				

Test ID	bac-per-002	Category	Performance	Severity	Major
Objective	To test the get file content functionality's performance				
Steps	1. Perform a get file content functionality REST API request in Postman as in the Test ID: back-fun-002 2. Denote the time between the request and the response				

Expected	The response for the get file content REST API request must be visible in Postman within 5 seconds.
Date-Result	04/2023 -?

Test ID	bac-sec-001	Category	Security	Severity	Major
Objective	To test the security of the user functionalites in the REST API such as upload file, edit user information, delete account and so on.				
Steps	1. Prepare an upload file POST request in the Postman and add the parameter file_content with its' key filename in the body of the request 2. Send the POST request without adding the token in the token authentication part 3. Prepare a delete account functionality DELETE request in the Postman 4. Send the request without adding the token in the authentication part in the request				
Expected	The responses should raise an error without returning any response because of the token being non-existent because of the security issues.				
Date-Result	04/2023 -?				

Test ID	bac-fun-003	Category	Functional	Severity	Major
Objective	To test the get all file information functionality in REST API				
Steps	1. Send a sign in request like the Test Id: back-fun-001 2. Get the token in the response 3. Prepare a GET request in Postman with the following link: https://neutlan.com/api/user/get_all_file_info 4. Apply the token in the authentication part of the body in the get_all_info request 5. Send the GET request 6. Denote the response				

Expected	The response should be an array names files_info with the all file information of the particular user and the response should have the value 200.
Date-Result	04/2023 -?

Test ID	bac-per-003	Category	Performance	Severity	Major
Objective	To test the get all file information functionality's performance				
Steps	1. Send a get_all_file_info request via Postman like the Test ID: back-fun-003 2. Denote the time between the sending request and getting response				
Expected	The response for the get all file information REST API request must be visible in Postman within 5 seconds.				
Date-Result	04/2023 -?				

Test ID	bac-fun-004	Category	Functional	Severity	Major
Objective	To test the sign up functionality in REST API				
Steps	1. Prepare a POST request to the REST API with the link: https://neutlan.com/api/auth/sign_up in Postman 2. Add the parameters as email, password, and password_confirm in the request for a user 3. Send the POST request via Postman 4. Denote the response as the successful sign up				
Expected	The response should be the value: 200 HTTP response.				
Date-Result	04/2023 -?				

Test ID	bac-per-004	Category	Performance	Severity	Major
Objective	To test the sign up functionality's performance				
Steps	1. Send a sign up functionality request as in the TEST ID: back-fun-004 2. Denote the time between the request and the response				
Expected	The response for the REST API request must be visible in Postman within 5 seconds.				
Date-Result	04/2023 -?				

Test ID	bac-fun-005	Category	Functional	Severity	Major
Objective	To test the delete account functionality in REST API				
Steps	1. Send a sign in request like the Test Id: back-fun-001 2. Get the token in the response 3. Prepare a POST request in Postman with the following link: https://neutlan.com/api/user/delete_account 4. Apply the token in the authentication part of the body in the delete_account request 5. Add the password for the current user as parameter into the body 7. Send the POST request 8. Denote the response				
Expected	The response should have the value 200.				
Date-Result	04/2023 -?				

Test ID	bac-per-005	Category	Performance	Severity	Major

Objective	To test the delete account functionality's performance
Steps	1. Send a delete_account request via Postman like the Test ID: back-fun-005 3. Denote the time between the sending request and getting response
Expected	The response for the delete_account REST API request must be visible in Postman within 5 seconds.
Date-Result	04/2023 -?

Test ID	bac-fun-006	Category	Functional	Severity	Moderate
Objective	To test the edit profile functionality in REST API				
Steps	1. Send a sign in request like the Test Id: back-fun-001 2. Get the token in the response 3. Prepare a POST request in Postman with the following link: https://neutlan.com/api/user/edit_profile_info 6. Apply the token in the authentication part of the body in the edit_profile_info request 7. Add the edited name, email and password of the user as parameter in the body of the request 9. Send the POST request 10. Denote the response				
Expected	The response should have the value 200.				
Date-Result	04/2023 -?				

Test ID	bac-per-006	Category	Performance	Severity	Moderate
Objective	To test the edit profile functionality's performance				

Steps	1. Send a edit_profile_info request via Postman like the Test ID: back-fun-006 2. Denote the time between the sending request and getting response
Expected	The response for the edit_profile_info REST API request must be visible in Postman within 5 seconds.
Date-Result	04/2023 -?

6. Consideration of Various Factors in Engineering Design

During the research and implementation processes, there are several factors that may have either a small or significant impact to our project NEUTLAN. These factors can be described as follow.

6.1. Public Safety

NEUTLAN will detect the gender biased words and phrases for those who will be using it. By eliminating them, it will enable users to establish healthy communication and prevent people from being offended in case of an usage of those words or phrases accidently.

6.2. Individual's Privacy

According to the choice of the users, the data will be collected. However, that data should not be linkable to the user. Hence, the gathering of the data has to be anonymous in

order to ensure the privacy of individuals. Also, for those who will not allow their data to be sent, their choices will be respected and their data will not be collected.

6.3. Global Factors

NEUTLAN is expected to be published for global usage since English is used widely and usage of gender-biased language is a global problem.

6.4. Social Factors

NEUTLAN will create social awareness by helping to reduce gender-biased language usage. Due to the many Chrome browser users and the convenience of using extensions in it, many audiences will be reached.

6.5. Economic Factors

NEUTLAN will not pursue any profit by providing this service to the users. Hence, the registration and usage of this application will be free. However, the costs of the implementation and deployment of the application will be paid by the developers.

The factors which are described above might cause the changes in our analysis and design due to some unforeseen problems. The problems and their effect on our analysis and design are summarized in the table given below. The scale for each factor is also set to put the level of importance.

Table 1: Factors that can affect design and its importance

	Effect level

Public Safety	10
Individual’s Privacy	7
Global Factors	6
Social Factors	10
Economic Factors	7

7. Teamwork Details

7.1. Contributing and functioning effectively on the team

As the team responsible for carrying out the Neutlan project, we have adjusted well to working together as a group, thanks to previous projects that we have completed together. Each member of our group has different interests and is well-trained in specific topics. When it comes to work participation, we take into consideration these interests and capabilities. Now, we will briefly talk about each member's contribution to the project;

Sıla Saraoğlu: Sıla is responsible for our project's backend-related technologies. She has been trained and worked as a backend developer both in her internships and other projects that we have completed.

Berke Ceran: Berke is leading our front-end-related work. He is also dealing with frontend for a while both in his internships and also in other projects that we have done together.

Lara Merdol: Lara is dealing with the NLP model and also finding appropriate datasets for training the model. She has worked on similar projects before and with the help of Akmuhammet she improved herself in that subject.

Akmuhammet Ashyralyyev: Akmuhammet functions as the backbone of the project. He is responsible for leading the project process and helping others when there is a need. He also managed our project DevOps-related work such as opening a remote server and its configuration.

7.2. Helping create a collaborative and inclusive environment

Even though each of us leads a different part of the project, we always share information and discuss points that we struggle with together. We always try to stay updated and informed about each other's work. Since the beginning of the semester, we meet at least twice a week, in addition to our social gatherings, to discuss our current state and plan for the future. In our project, we strive to be as professional as possible, and for this purpose, we regularly use project management tools such as Github, Confluence, and others. Good communication and friendship are some of our best advantages, as it saves us time and enables us to complete our project without facing critical problems that could affect our project. Every member of our group is aware of their responsibility in the project and tries to be collaborative as possible.

7.3. Taking a lead role and sharing leadership on the team

Collaboration and effective teamwork are critical components of any successful project. As mentioned in section 7.1, we have adopted a work allocation system that allows each team member to lead a specific part of the project. This strategy has been demonstrated to be highly effective in optimizing time and resources. By assigning each person a specific area of expertise, we can leverage our strengths and competencies to the fullest extent. Each team member can focus on their assigned tasks and take ownership of the process, which ultimately leads to better outcomes. As a result, the project progresses smoothly and efficiently, without any unnecessary delays or complications. To ensure that everyone is on the same page and that progress is being made, we hold weekly progress meetings. During these meetings, each team member provides an update on the progress or problems they are encountering in their part of the project. We then brainstorm together to come up with solutions to any issues that arise. Additionally, we have established a culture of open communication and collaboration, where everyone is encouraged to share their ideas and be receptive to feedback. This ensures that all team members are actively engaged and invested in the project's success.

8. Glossary

Term	Definition
GFL	Gender-fair language (GFL) aims at reducing gender stereotyping and discrimination [6].
NLP	Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data [7].
TensorFlow	TensorFlow is an open-source library that is free to use for machine learning and artificial intelligence. It can be utilized for various tasks, but its primary emphasis is on deep neural network training and inference [8].
Mongo DB	MongoDB is a cross-platform document-oriented database application that is open source. MongoDB, a NoSQL database application, employs documents that resemble JSON and may or may not include schemas [9].

Amazon Lightsail Service	Amazon Lightsail is a virtual private server (VPS) service provided by Amazon Web Services (AWS) that offers a simple way for developers, small businesses, and individuals to host their applications and websites in the cloud [10].
Django	Django is an advanced Python web framework that encourages fast development [11].
Postman	Postman is an API platform for building APIs and testing the APIs created for the development purposes [12].
Confluence	Confluence is a team workspace for creating, collaborating, and organizing all of the project work such as documentation in one place which is developed by Atlassian [13].
Google Colab	Google Colab is a data analysis and machine learning tool lets you to combine Python code and rich text to be stored as a single document in Google Drive [14].

9. References

[1] B. Baschuk, “Global gender gap: WEF says it's narrowing but will take 132 years to get parity,” Bloomberg.com, 13-Jul-2022. [Online]. Available: https://www.bloomberg.com/news/articles/2022-07-13/wef-gender-gap-narrows-after-pandemic-delayed-parity-by-decades?leadSource=uverify+wall [Accessed: 16-Oct-2022].

[2] S. Sczesny, M. Formanowicz, and F. Moser, “Can gender-fair language reduce gender stereotyping and discrimination?,” Frontiers. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2016.00025/full [Accessed: 16-Oct-2022].

[3] A. Madgavkar, “A conversation on Artificial Intelligence and gender bias,” McKinsey & Company, 01-Aug-2022. [Online]. Available: https://www.mckinsey.com/featured-insights/asia-pacific/a-conversation-on-artificial-intelligence-and-gender-bias [Accessed: 16-Oct-2022].

[4] “Your ultimate writing resource,” *Grammarly Blog*. [Online]. Available: https://www.grammarly.com/blog/. [Accessed: 10-Mar-2023].

[5] D. Adewusi, “Everything You Need to Know about Google Translate,” *Scientific Editing*. [Online]. Available: https://www.scientific-editing.info/blog/everything-you-need-to-know-about-google-translate/. [Accessed: 10-Mar-2023].

[6] M. Berger, “A guide to how gender-neutral language is developing around the world,” *The Washington Post*, 15-Dec-2019. [Online]. Available: https://www.washingtonpost.com/world/2019/12/15/guide-how-gender-neutral-language-is-developing-around-world/. [Accessed: 12-Mar-2023].

[7] “What is Natural Language Processing (NLP)?,” *Educative*. [Online]. Available: https://www.educative.io/answers/what-is-natural-language-processing-nlp. [Accessed: 12-Mar-2023].

[8] “Why tensorflow,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/about. [Accessed: 12-Mar-2023].

[9] “What is mongodb?,” *Educative*. [Online]. Available: https://www.educative.io/answers/what-is-mongodb? [Accessed: 12-Mar-2023].

[10] “AWS documentation.” [Online]. Available: https://docs.aws.amazon.com/. [Accessed: 12-Mar-2023].

[11] Django, “The Web framework for perfectionists with deadlines | Django,” *Djangoproject.com*, 2019. https://www.djangoproject.com/ [Accessed: 12-Mar-2023].

[12] Postman, “Postman | The Collaboration Platform for API Development,” *Postman*, 2021. https://www.postman.com/ [Accessed: 12-Mar-2023].

[13] Atlassian, “Confluence: A brief overview,” *Atlassian*. [Online]. Available: https://www.atlassian.com/software/confluence/guides/get-started/confluence-overview. [Accessed: 12-Mar-2023].

[14] “Colaboratory,” *Google Workspace Marketplace*. [Online]. Available: https://workspace.google.com/marketplace/app/colaboratory/1014160490159. [Accessed: 12-Mar-2023].