A Project Report

On

Hate Speech Detection Model

Submitted in partial fulfillment of the

requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

Computer Science Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Session 2023-24

in

Machine Learning (NLP)

By

Harsh Vyas 21SCSE1010212

Abhimanyu Singh 21SCSE1010287

Ashutosh Mishra 21SCSE1010696

Under the guidance of

Ajay Sikandar

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA

Nov, 2023

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"……"** in partial fulfillment of the requirements for the award of the B. Tech. (Computer Science and Engineering) submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of Feburary, 2023 to May and 2023, under the supervision of Prof. SantoshKumar, Department of Computer Science and Engineering, of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Student Names (Admission No.)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Guide Names

Designation

CERTIFICATE

This is to certify that Project Report entitled "..... ……..…." which is submitted by ……..... in partial fulfillment of the requirement for the award of degree B. Tech. in Department of ..... of School of Computing Science and Engineering Department of Computer Science and Engineering

Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Program Chair **Signature of Dean**

Date:

Place: Greater Noida

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Professor ……., Department of Computer Science & Engineering, Galgotias University, Greater Noida, India for his constant support and guidance throughout the course of our work. His/Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Professor (Dr.) ……., Head, Department of Computer Science & Engineering, Galgotias University, Greater Noida, India for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:

Name :

Roll No.:

Date :

Signature:

Name :

Roll No.:

Date :

Signature:

Name :

Roll No.:

Date :

ABSTRACT

Surfing on internet nowadays can lead to exposure to a lot of information which might not be suitable for the person, majorly children. This model aims at blocking explicit words, hate and racist comments on various websites, blogs, online web games chat box. There are many apps that have inbuilt safe search options which blocks hatred, racist and abusive words in few known languages. This model allows the user to add an extension in their web browser which blocks some existing hate speech words which are stored in a dataset and also lets the user to add custom words and slangs in their native language. This feature of blocking custom words in any language is not available anywhere else. Parents or guardians can take control over what their children are getting exposed to and can create a safe environment for them. This problem needs addressing as we personally have faced a lot of issues regarding hate speech and we did not have proper knowledge as to what these things were. On getting introduced to explicit words without any prior knowledge we further researched about it out of curiosity leading to half knowledge about many things. So, we want to prevent exposure to half knowledge and make surfing on internet safer.

TABLE OF CONTENTS

Page

DECLARATION ii

CERTIFICATE……..... iii

ACKNOWLEDGEMENTS iv

ABSTRACT v

CHAPTER 1

1.1 7

1.2 7

1.3 8

CHAPTER 2

2.1. 9

2.2. 15

2.3. 16

2.4. 17

2.5. 17

2.6. 18

2.7. 23

CHAPTER 3

3.1. 25

3.2. 25

3.1. 26

3.2. 26

3.1. 26

3.2. 27

3.2. 27

CHAPTER 4

4.1. 28

4.2. 29

CHAPTER 5 32

REFERENCES…….…… 34

CHAPTER 1
INTRODUCTION

In the current digital era, hate speech is a widespread problem that has the ability to hurt people, propagate discrimination, and have a harming effect on both communities and individuals. This research intends to construct an advanced hate speech detection model utilizing cutting-edge natural language processing (NLP) techniques in response to the growing concern around online hate speech.

Online platforms now face a serious threat from hate speech, which is defined as any writing, conduct, speech, or expression that disparages, threatens, or offends an individual or group because of characteristics including race, ethnicity, religion, gender, sexual orientation, handicap, or nationality. The capacity to recognize and eliminate hate speech automatically is essential to creating a more secure and welcoming online community

1.1 Motivation

Thanks to social media, blogs, online games, and other platforms, children of all ages can now access the digital world. Young children may be exposed to information that is inappropriate for them at that age. The identical problems that we had encountered as small children. After being introduced to this information, we started researching it online and learned only a portion of it, which is extremely risky. Confusion and misinterpretation of certain significant life phases(puberty) result from this as well. After witnessing this, we were inspired to develop this approach, which can filter offensive language and hate speech to provide a secure atmosphere for young children in particular.

1.2 Project Objective

The hate speech detection model's main goal is to precisely locate and classify hate speech instances in text data. To distinguish between hate speech and non-hateful information, this entails creating a strong machine learning model that can examine linguistic patterns, contextual clues, and other characteristics.

The project's main objectives are as follows:

a) Accuracy and Precision: Create a model with few false positives and high accuracy in detecting hate speech. To make sure that appropriate content is not mistakenly labeled as hate speech, accuracy is essential.

b) Multilingual Support: Make sure the model can identify hate speech in a variety of languages, which will make it an effective tool for combating hate speech globally.

c) Customization for Particular Domains: Recognize that hate speech can take on diverse forms in different circumstances and allow the model to be customized to meet the needs of particular businesses, groups, or domains.

d) Privacy Considerations: Put in place safeguards to preserve user privacy by eliminating pointless data collecting and making sure that applicable privacy laws are followed.

This study lays the groundwork for the creation and application of a hate speech detection model that complies with moral principles and the more general objective of creating a safer online environment by outlining the motivation, project objectives, and scope.

1.3 Scope of the Project

The extension will scan web page content for predefined hate speech words and phrases, preventing their display to the user. Users can add, edit, or remove custom words or phrases to supplement the hate speech blocking functionality. Develop an intuitive interface accessible through the browser toolbar or settings panel for easy management of settings and word lists.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

2.1 Product Perspective

The Hate Speech and Custom Word Blocking Extension is a browser-based tool developed to enhance online safety by preventing the display of hate speech terms and enabling users to add their own custom words for blocking. It operates as an add-on for major web browsers, providing users with control over their browsing experience by promoting a more respectful and inclusive online environment.

The following subsections describe how the software operates inside various constraints.

2.1.1 System Interfaces

The Hate Speech and Custom Word Blocking Extension interfaces with several components to fulfill its functionalities:

Browser API Interface: The extension interacts with the browser's API to access and modify web page content. It utilizes browser-provided functionalities for content scanning, identification of hate speech words, and blocking mechanisms within the webpage environment.

User Interface (UI): The extension incorporates a user-friendly graphical interface accessible through the browser toolbar or settings panel. This UI allows users to manage settings, enable/disable hate speech blocking, and add/edit custom word lists effortlessly.

Storage Interface: Manages the storage and retrieval of hate speech word lists, custom word entries, and user preferences within the extension. It ensures the persistence of user-defined settings and custom word lists across browsing sessions.

2.1.2 Interfaces

User Interface (UI):

Visual Clarity: Intuitive design with clear and easily understandable elements for managing hate speech and custom word lists.

Usability: Simple navigation and intuitive controls for adding, editing, and removing custom words without technical complexities.

Feedback Mechanisms: Provide real-time feedback on word addition/deletion and enable easy toggling of hate speech blocking.

Accessibility: Ensure compatibility with various screen sizes, readability, and support for assistive technologies.

Browser Interface:

Efficiency: Optimize API calls and data handling for seamless integration with the browser environment.

Reliability: Ensure consistent and accurate scanning of web page content for hate speech detection without impacting browsing speed.

Security: Adhere to browser security standards to prevent unauthorized access or manipulation of browser data.

Storage Interface:

Data Integrity: Reliable storage and retrieval of hate speech word lists, custom words, and user preferences without loss or corruption.

Efficient Retrieval: Quick access and retrieval of stored data to support rapid processing and updating of word lists.

Privacy Measures: Implement encryption or secure storage mechanisms to protect user-specific data.

2.1.3 Hardware Interfaces

1. System Requirements:

Computer or Device: The extension runs on various devices compatible with modern web browsers (desktops, laptops, tablets, mobile phones).

Processor and Memory: Basic hardware configurations suitable for running the chosen browser smoothly.

Storage Space: Adequate storage for the browser and extension files, although the extension itself doesn't consume significant storage space.

2. Accessibility Devices:

Compatibility with hardware accessibility devices like screen readers, braille displays, or alternative input devices for users with disabilities.

3. Internet Connectivity:

Stable internet connectivity for initial installation, updates, and real-time detection of web content.

2.1.4 Software Interfaces

(1) **Name:** Microsoft Visual Studio Code
(2) **Mnemonic:** VS Code
(3) **Specification number:** APP01
(4) **Version number:** 1.85
(5) **Source:** Microsoft Corporation

VS Code provides us with an interface that allows us to work on our code and also to prepare the JSON file that is required to define the properties of the web extension.

With this we can work on multiple coding languages and tools all in a single platform. And VS Code is also known for its regular updates and bug fixes that allows developers to take work ignoring the external factors.

2.1.5 Communications Interfaces

The Hate Speech and Custom Word Blocking Extension primarily communicates within the browser environment and with external servers or databases for updates. Its communication interfaces include:

Browser-Internal Communication: Interacts with the browser's APIs for scanning web content and managing user interface interactions.

External Server Communication: Periodically communicates with external servers or databases to update predefined hate speech word lists, ensuring the extension remains current and effective.

User Feedback: Provides real-time feedback within the extension's interface, indicating successful actions (addition/removal of custom words) or detection of blocked content.

Analytics Reporting: Optionally, communication with analytics tools to collect usage data for improving functionality and user experience.

2.1.6 Memory Constraints

There are no such memory constraints for this project. The minimum required specifications are the same as that of the web browser being used by the user. Until and unless the machine is capable to run the web browser which is the general case, our extension will work smoothly.

2.1.7 Operations

1. Hate Speech Detection Operations:

a. Scanning:

Content Analysis: Scans web page content for predefined hate speech words and phrases.

Algorithm Execution: Executes hate speech detection algorithms within the content.

b. Blocking:

Prevention Mechanism: Blocks the display of identified hate speech words or phrases on visited web pages.

2. User Interaction Operations:

a. Custom Word Management:

Addition: Enables users to input custom words or phrases for blocking.

Modification: Allows users to edit existing custom entries.

Deletion: Enables users to remove custom words from the blocking list.

b. Interface Interaction:

Settings Toggle: Allows users to activate or deactivate hate speech blocking.

Settings Management: Provides a user-friendly interface for managing extension settings and custom word lists.

3. Data and Storage Operations:

a. Data Retrieval:

Retrieving Word Lists: Accesses predefined hate speech word lists.

Custom Word Access: Retrieves and manages user-defined custom word lists and settings.

b. Storage Management:

Saving Changes: Saves modifications made to hate speech word lists or custom word entries.

Persistence: Ensures stored data (word lists, custom entries, user settings) persists between browser sessions.

4. System Maintenance Operations:

a. Updates and Maintenance:

Word List Updates: Periodically updates predefined hate speech word lists to stay current.

Extension Updates: Ensures the extension remains compatible with browser updates.

5. Feedback and Alert Operations:

a. User Feedback:

Real-time Alerts: Provides immediate feedback on successful addition/removal of custom words or detection and blocking of hate speech content.

6. Performance Optimization Operations:

a. Resource Management:

Efficient Resource Usage: Manages memory and system resources to optimize performance without causing excessive consumption.

2.1.8 Site Adaptation Requirements

Site adaptation refers to the extension's ability to adapt and function optimally across various websites and web environments. For the Hate Speech and Custom Word Blocking Extension, site adaptation requirements are crucial for its effective performance. Here are the key site adaptation requirements:

1. Compatibility with Different Websites:

Universal Compatibility: Ensure the extension functions across diverse websites, including social media platforms, news sites, forums, and other web content.

Adaptation to Web Structures: Accommodate variations in web page structures, different HTML elements, and content layouts without compromising detection accuracy.

2. Sensitivity Adjustment:

Website Sensitivity Settings: Allow users to adjust hate speech detection sensitivity levels on different websites based on varying content structures and contexts.

3. Exception Handling:

Exception Rules: Enable users to create exceptions for specific websites where hate speech blocking might interfere with legitimate content or discussions.

4. Performance Optimization for Different Websites:

Efficiency across Pages: Optimize hate speech detection algorithms to perform consistently across various websites while minimizing false positives or negatives.

5. Real-Time Adaptation:

Dynamic Detection: Ensure the extension dynamically adapts to changes in website content in real-time without requiring manual adjustments.

6. Compatibility with Browser Updates:

Browser Compatibility: Remain compatible with different browser versions and updates without compromising functionality across varied web environments.

7. User-Friendly Interface for Adaptation:

Accessible Settings: Provide an intuitive user interface for adjusting sensitivity levels and creating exceptions for specific websites.

Clear Instructions: Offer clear instructions or tooltips guiding users on how to optimize settings for different websites.

8. Continuous Improvement:

Feedback Mechanism: Collect user feedback to identify adaptation issues and improve site compatibility based on user experiences across various websites.

2.2 Product Functions

1. Hate Speech Detection and Blocking:

a. Scanning Functionality:

Content Analysis: Scans web page content in real-time for predefined hate speech words and phrases.

Algorithm Execution: Executes hate speech detection algorithms to identify offensive language.

b. Blocking Mechanism:

Prevention of Display: Blocks the display of detected hate speech words or phrases on visited web pages.

2. Custom Word Management:

a. User Input Functions:

Addition of Custom Words: Allows users to input custom words or phrases for blocking.

Modification and Deletion: Enables editing and removal of custom entries from the blocking list.

3. User Interface Functions:

a. Settings Management:

Activation and Deactivation: Provides options to activate or deactivate hate speech blocking functionality.

Settings Accessibility: Offers an intuitive interface for managing extension settings and custom word lists.

4. Data and Storage Functions:

a. Retrieval and Storage:

Access to Hate Speech Word Lists: Retrieves predefined hate speech word lists.

Storage of Custom Entries: Manages storage of user-defined custom word lists and settings.

5. System Maintenance Functions:

a. Updates and Compatibility:

Word List Updates: Periodic updates to predefined hate speech word lists to ensure relevance.

Extension Updates: Maintains compatibility with browser updates for seamless operation.

6. Feedback and Alert Functions:

a. User Feedback Mechanisms:

Real-time Alerts: Provides immediate feedback on successful addition/removal of custom words or detection and blocking of hate speech content.

7. Performance Optimization Functions:

a. Resource Management:

Efficient Resource Usage: Manages memory and system resources for optimal performance without excessive consumption.

2.3 User Characteristics

1. **Diverse Backgrounds:** Users from different age groups, ethnicities, cultures, and geographic locations.
2. **Frequent Internet Users:** Individuals who regularly browse the internet, including social media, news sites, forums, and various online platforms.
3. **Safety and Inclusivity Concerns:** Users concerned about encountering hate speech, seeking a safer online experience, and supporting inclusive and respectful online interactions for all users.
4. **Customization and Control Seekers:** Users who prefer customization options in managing their online experience, seeking flexibility and control over hate speech blocking preferences.
5. **Varied Technical Skills:** Tech-savvy users comfortable with technology and novices seeking simplicity and user-friendly interfaces within the extension.
6. **Privacy-Conscious Users:** Individuals wary of sharing personal information, seeking assurances of data security and privacy within the extension.
7. **Accessibility Needs:** Users relying on assistive technologies requiring accessible interfaces for seamless interaction with the extension.
8. **Efficiency Seekers:** Users looking for efficient and effective tools that enhance their browsing experience without interruptions.

2.4 Constraints

Constraints within the Hate Speech and Custom Word Blocking Extension project encompass various limitations and considerations that can impact its design, functionality, and performance.

1. **Browser API Limitations:** The extension's functionality might be limited by constraints within the browser's APIs, affecting the depth and efficiency of content scanning and manipulation.
2. **Resource Constraints:**

   Memory and Processing Power: Limited memory or processing capabilities of user devices can impact the extension's performance, requiring efficient resource management.

Storage Space: Restrictions on storage capacity may limit the amount of data the extension can store locally.

3. **Compatibility Issues:** Differences in browser versions or platforms might affect the extension's operation, requiring extensive testing and adaptation for multiple browsers.
4. **Data Privacy Regulations:** Adherence to stringent data privacy regulations might limit data collection, storage, or transmission practices within the extension.
5. **Language and Context Limitations:** Challenges in accurately detecting hate speech due to varying contexts, languages, and evolving language patterns can impact detection accuracy.
6. **Continuous Maintenance:** Regular updates and maintenance to keep up with evolving hate speech terminology and browser updates can pose ongoing challenges.
7. **User Adoption and Acceptance:** Potential resistance or challenges in educating users about the extension's features, usage, and privacy practices can impact adoption rates.
8. **Performance Impact**: Heavy resource consumption by the extension might impact overall browser performance or cause conflicts with other installed extensions.
9. **Accessibility Considerations:** Ensuring compatibility with various assistive technologies to address accessibility needs can be a constraint.

2.5 Assumptions and Dependencies

Assumptions:

1. **Stable Browser Environments:** The extension assumes stable and consistent browser environments across various platforms and versions for proper functionality.
2. **Consistent API Functionality:** Assumes consistent and reliable functionality of browser APIs for content scanning and manipulation.
3. **User Awareness:** Assumes users have a basic understanding of extension installation and usage procedures.
4. **Internet Connectivity:** Relies on users having stable internet connectivity for initial installation and periodic updates of predefined word lists.
5. **Language and Context Accuracy:** Assumes accuracy in detecting hate speech based on predefined word lists, considering variations in language and contextual nuances.

Dependencies:

1. **Browser APIs:** Relies on the functionality and support of browser APIs for content scanning and manipulation within the web page environment.
2. **Data Updates:** Dependencies on external sources or servers for regular updates to predefined hate speech word lists to maintain relevance and accuracy.

3. **System Resources:** Dependent on the availability of sufficient system resources (memory, storage) for optimal performance.
4. **Compliance Standards:** Dependencies on adherence to data privacy regulations and standards for handling user data and privacy concerns.
5. **User Acceptance:** The success of the extension depends on user acceptance, understanding, and willingness to utilize and engage with the extension's functionalities.
6. **Browser Compatibility:** Dependent on consistent compatibility with different browser versions and platforms for seamless operation across diverse user environments

2.6 Apportioning of Requirements.

Apportioning of requirements involves dividing and prioritizing features or functionalities of a project across different phases or iterations based on various considerations. In the context of the Hate Speech and Custom Word Blocking Extension, the apportioning of requirements can be delineated as follows:

Phase-wise Allocation:

1. Core Functionality Development:

Phase 1 - Core Features:

Implement hate speech detection algorithms and basic blocking functionality.

Develop user interface for adding custom words and activating/deactivating the extension.

2. Enhancements and Customization:

Phase 2 - User Customization:

Introduce settings management to enable users to adjust sensitivity levels and manage exceptions.

Enhance the UI for better user experience and intuitive interaction.

3. Performance Optimization and Maintenance:

Phase 3 - Performance Optimization:

Focus on optimizing resource usage to improve extension performance without compromising functionality.

Implement mechanisms for efficient data storage and retrieval.

4. Advanced Features and Adaptation:

Phase 4 - Advanced Adaptation:

Work on adapting the extension to diverse websites and contexts for better compatibility.

Implement advanced language processing for improved hate speech detection accuracy.

Iterative Development Approach:

Iteration 1 - Minimum Viable Product (MVP):

Develop and release a basic version focusing on fundamental hate speech detection and blocking functionalities.

Iterations 2 and Beyond - Feature Iterations:

Introduce subsequent iterations with incremental enhancements based on user feedback, adding customization options, addressing performance issues, and adapting to different web environments.

Risk-Based Allocation:

High-Risk Mitigation:

Allocate resources to address high-risk areas like data privacy compliance, browser compatibility, and user acceptance early in the development process.

Market and User Research Allocation:

Research and Feedback Incorporation:

Allocate resources periodically for market research and user feedback integration to align the extension with evolving user needs and industry trends.

Regulatory Compliance Allocation:

Adherence to Regulations:

Allocate resources to ensure ongoing compliance with evolving data privacy regulations and standards throughout the development life cycle.

2.7. Use case

2.7.1. Use case Model

The use case model for the Hate Speech and Custom Word Blocking Extension outlines the interactions between the users and the extension, detailing various scenarios and functionalities. Here are some sample use cases for the extension:

Use Case 1: Blocking Hate Speech

Actors:

User

Description:

Scenario: The user visits a social media platform.

Trigger: The user activates the hate speech blocking feature.

Flow:

1. The extension scans the webpage content.
2. It detects hate speech words/phrases.
3. The extension blocks the display of identified hate speech content.

Outcome: The user's browsing experience is free from displayed hate speech content.

Use Case 2: Adding Custom Words

Actors:

User

Description:

Scenario: The user encounters an offensive word not covered by default settings.

Trigger: The user accesses the extension settings.

Flow:

1. The user navigates to the custom word management section.
2. Adds the offensive word to the custom word list.

Outcome: The custom word is now part of the blocking mechanism, preventing the display of the added offensive word.

Use Case 3: Adjusting Sensitivity Settings

Actors:

User

Description:

Scenario: The user finds certain websites overly sensitive or not sensitive enough.

Trigger: The user accesses the extension settings.

Flow:

1. The user navigates to sensitivity settings.
2. Adjusts the sensitivity level for a specific website.

Outcome: The extension adapts its hate speech detection on the chosen website according to the adjusted sensitivity level.

Use Case 4: Managing Exceptions

Actors:

User

Description:

Scenario: The user wants to exempt certain websites from hate speech blocking.

Trigger: The user accesses the extension settings.

Flow:

1. The user navigates to the exceptions section.
2. Adds specific websites to the exception list.

Outcome: Hate speech blocking is disabled on the exempted websites while active on other sites.

Use Case 5: Updating Hate Speech Word List

Actors:

User

Extension

Description:

Scenario: Periodic updates to predefined hate speech word lists.

Trigger: Scheduled or triggered updates.

Flow:

1. The extension connects to an external server for updates.
2. Retrieves the updated hate speech word list.

Outcome: The extension's predefined word list is updated, enhancing accuracy in hate speech detection.

The use case model for the Hate Speech and Custom Word Blocking Extension illustrates various scenarios where users interact with the extension to block hate speech, customize settings, and manage exceptions, facilitating a safer and personalized browsing experience.

2.7.2 Use Case Diagram/Scenario

Use Case Diagram: Hate Speech and Custom Word Blocking Extension

Actors:

User

Use Cases:

1. Block Hate Speech

2. Add Custom Words

3. Adjust Sensitivity Settings

4. Manage Exceptions

5. Update Hate Speech Word List

User -> Block Hate Speech

User -> Add Custom Words

User -> Adjust Sensitivity Settings

User -> Manage Exceptions

Use Case Number	Application	Use Case Name	Use Case Description	Primary Actor	Precondition	Trigger	Basic Flow	Alternate Flows
1	Extension	Block Hate Speech	Prevents the display of hate speech content on visited web pages	User	Extension installed and active	User activates hate speech blocking feature	1. User enables hate speech blocking. 2. Extension scans web page content. 3. Detected hate speech is blocked.	-
2	Extension	Add Custom Words	Allows users to add custom words or phrases for blocking	User	Extension installed and active	User accesses custom word management	1. User adds a custom word to the blocking list.	-
3	Extension	Adjust Sensitivity Settings	Enables users to modify sensitivity levels for hate speech detection	User	Extension installed and active	User accesses sensitivity settings	1. User adjusts sensitivity levels for a specific website.	-
4	Extension	Manage Exceptions	Allows users to exempt certain websites from hate speech blocking	User	Extension installed and active	User navigates to the exceptions section	1. User adds specific websites to the exception list.	-
5	Extension	Update Hate Speech Word List	Periodically updates predefined hate speech word lists	Extension	-	Scheduled or triggered updates	1. Extension retrieves updated speech word list.	-

Extension -> Update Hate Speech Word List

2.7.3 Use Case Scenario

Use Case Scenario 1: Blocking Hate Speech

Scenario: Sarah, an active social media user, wants to avoid encountering hate speech content while browsing her favourite platforms.

Primary Actor: Sarah (User)

Precondition: The extension is installed and activated in Sarah's browser.

Trigger: Sarah encounters hate speech content on a social media platform.

Basic Flow:

1. Sarah activates the hate speech blocking feature via the extension settings.
2. As Sarah browses her social media feed, the extension scans the content in real-time.
3. The extension detects hate speech words or phrases within the feed.
4. Detected hate speech content is blocked from display on Sarah's browser.

Outcome: Sarah can browse her social media platforms without encountering displayed hate speech content, fostering a safer and more comfortable browsing experience.

Use Case Scenario 2: Adding Custom Words

Scenario: Alex comes across an offensive term on a forum that is not covered by the extension's default settings.

Primary Actor: Alex (User)

Precondition: The extension is installed and active in Alex's browser.

Trigger: Alex encounters an offensive word not included in the predefined word list.

Basic Flow:

1. Alex accesses the extension settings and navigates to the custom word management section.
2. Alex adds the offensive word to the custom word list within the extension.
3. As Alex continues browsing, the extension now includes the added custom word in its hate speech detection mechanism.

Outcome: The extension now blocks the display of the newly added offensive word, contributing to a more tailored and refined hate speech detection system for Alex's browsing sessions.

CHAPTER3

SYSTEM DESIGN

The system design for the Hate Speech and Custom Word Blocking Extension involves various components and considerations to ensure effective functionality. Here's an outline of the system design:

3.1 Architecture:

1. **Browser Extension Framework:**

Develop using technologies compatible with major browsers (Chrome, Firefox, Edge, etc.).

Utilize browser-specific APIs for content scanning and manipulation.

2. **Client-Side Component:**

User Interface (UI) to manage extension settings, custom word lists, and sensitivity levels.

Background scripts handling content scanning, detection, and blocking mechanisms.

3.2 Functional Components:

1. **Content Scanning and Detection:**

Algorithms to scan web page content in real-time for hate speech words/phrases.

Predefined word lists for initial hate speech detection.

Language processing mechanisms for accuracy across diverse languages and contexts.

2. **Blocking Mechanism:**

Block the display of identified hate speech content on visited web pages.

Customizable settings to adjust sensitivity levels for detection.

3. **Custom Word Management:**

User-friendly interface to add, edit, or delete custom words or phrases for blocking.

4. **Settings Management:**

Interface allowing users to manage extension settings, exceptions, and sensitivity levels.

3.3 Data Management:

1. **Local Storage:**

Storage for custom word lists, user settings, and exceptions.

Efficient data management to ensure minimal impact on system resources.

2. **Server-Side Component (Optional):**

External server for periodic updates to predefined hate speech word lists.

3.4 Integration and Compatibility:

1. **Browser Compatibility:**

Ensure compatibility with different browser versions and platforms.

Testing across multiple browsers to ensure consistent functionality.

2. **API Integration:**

Utilize browser APIs for seamless integration and efficient content scanning.

3.5 Performance and Optimization:

1. **Resource Management:**

Optimize memory and processing usage to prevent excessive consumption.

Periodic maintenance to ensure efficient operation without compromising performance.

2. **Real-time Updates:**

Implement mechanisms for real-time updates of predefined hate speech word lists.

3.6 Security and Privacy:

1. **Data Encryption:**

Ensure sensitive data, such as custom word lists, is encrypted and securely stored.

Adherence to data privacy regulations and standards.

2. **Secure Communication:**

Secure communication between the extension and external servers (if applicable) for updates.

3.7 Testing and Maintenance:

1. **Quality Assurance:**

Thorough testing across different scenarios to ensure accurate hate speech detection.

User acceptance testing for usability and effectiveness.

2. **Continuous Improvement:**

Regular updates and maintenance to address bugs, enhance features, and adapt to evolving language patterns and user needs.

The system design focuses on creating a robust browser extension that effectively detects and blocks hate speech while offering customization options to users, ensuring compatibility, optimizing performance, maintaining security, and allowing for continuous enhancements and updates.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1. Software and Hardware Requirements

Software Requirements:

1. **Operating System:** Compatible with major operating systems like Windows, macOS, Linux.
2. **Browser Compatibility:** Supported on various browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, etc.
3. **Development Tools:**
    - IDEs (Integrated Development Environments) like Visual Studio Code, JetBrains WebStorm, or similar for extension development.
    - Frameworks and libraries suitable for extension development in JavaScript, HTML, CSS.
4. **Browser Extension APIs:** Utilization of browser-specific APIs for content scanning, manipulation, and UI components.

Hardware Requirements:

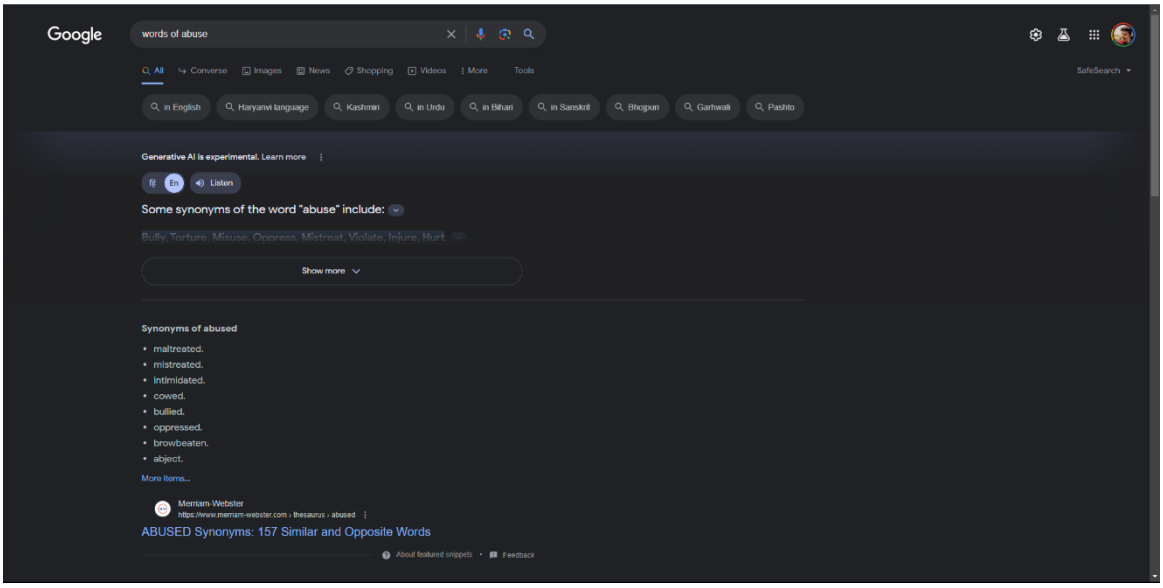
1. **Processor:** Standard processors compatible with the chosen operating system.
2. **Memory (RAM):** Moderate memory requirements based on extension complexity and resource optimization, typically ranging from 2GB to 4GB.
3. **Storage:** Minimal storage requirements for the extension itself, but may vary based on the volume of stored custom word lists and settings.
4. **Graphics:** No specific high-end graphics requirements, as it's a browser extension without extensive graphical needs.

Additional Considerations:

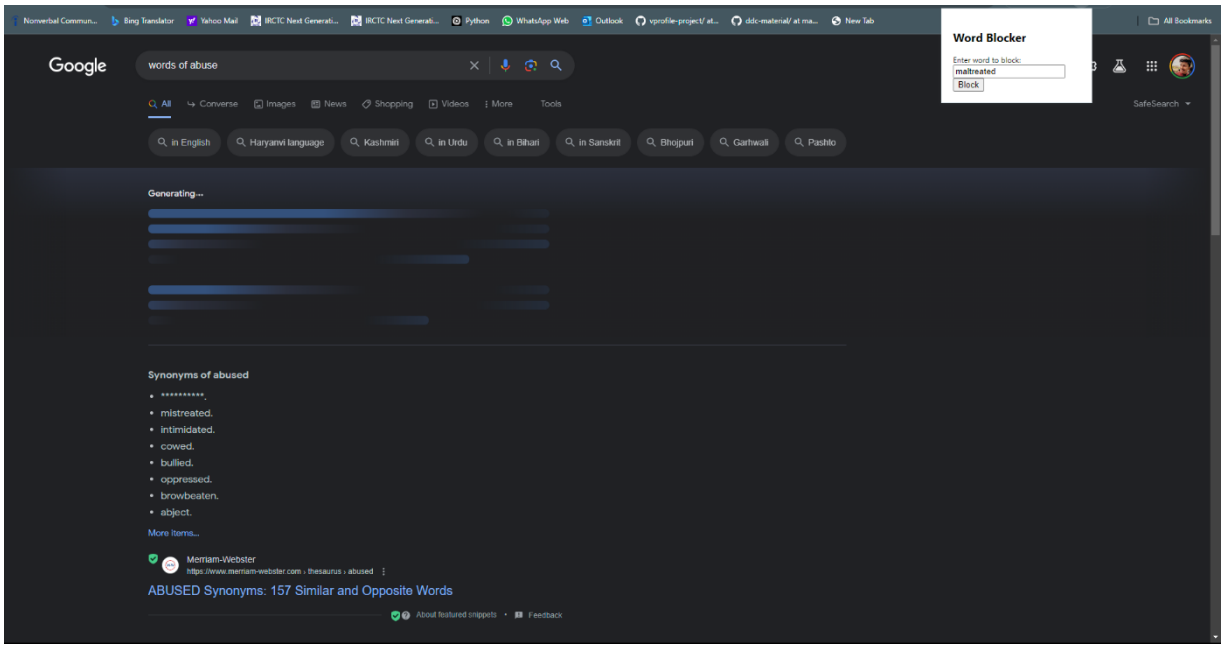
1. **Internet Connectivity:** Required for initial installation, updates, and potential retrieval of updated predefined hate speech word lists.
2. **Security Software Compatibility:** Ensure compatibility with various security software to avoid interference with extension functionality.
3. **Browser Updates:** Compatibility with different browser versions and ensuring the extension operates seamlessly with updated browsers.

4.2. Implementation Details

4.2.1. Snapshots Of Interfaces



(BEFORE ENABLING THE EXTENSION)



(AFTER USING THE EXTENSION)

4.2.2. Results

The Hate Speech and Custom Word Blocking Extension produces several key results aimed at fostering a safer and more personalized browsing experience for users. Here are the primary outcomes or results generated by this project:

1. Hate Speech Detection and Prevention:

- **Identification of Hate Speech:** Detection and blocking of predefined hate speech words and phrases on visited web pages.
- **Prevention of Display:** Blocking the display of identified hate speech content to users.

2. Customization and User Control:

- **Custom Word Management:** Ability for users to add, edit, or remove custom words or phrases for blocking.
- **Sensitivity Settings:** Options for users to adjust sensitivity levels for hate speech detection based on personal preferences.

3. Improved User Experience:

- **Safer Browsing Environment:** Creating a more secure and comfortable online environment for users by reducing exposure to hate speech content.
- **Personalized Settings:** Allowing users to personalize their hate speech blocking preferences and exceptions for a tailored experience.

4. Accessibility and Compatibility:

- **Compatibility Across Platforms:** Ensuring the extension works across various browsers and operating systems.
- **Accessibility Features:** Incorporating features for compatibility with assistive technologies for users with accessibility needs.

5. Maintenance and Updates:

- **Regular Updates:** Periodic updates to predefined hate speech word lists for relevance and accuracy.
- **Bug Fixes and Enhancements:** Continuous maintenance to address bugs, enhance features, and adapt to evolving language patterns and user needs.

6. User Education and Awareness:

- **Educational Material:** Providing information and instructions for users on the extension's functionalities, settings, and privacy practices.
- **Promotion of Safer Online Interactions:** Raising awareness about the importance of respectful online interactions and hate speech prevention.

CHAPTER 5

CONCLUSION

The conclusion of the Hate Speech and Custom Word Blocking Extension project encapsulates the achievements, impact, and future considerations after its development and implementation. Here's a summary of the project's conclusion:

Achievements and Impact:

1. **Functionality Implementation:** Successful development and deployment of an extension capable of detecting and blocking hate speech content.
2. **Customization Features:** Inclusion of user-controlled features allowing customization of hate speech blocking preferences.
3. **Enhanced User Experience:** Improved browsing experience by creating a safer online environment for users.
4. **Adaptability and Compatibility:** Compatibility across multiple browsers and operating systems, ensuring wider accessibility.
5. **Continuous Improvement:** Establishment of a foundation for ongoing updates, enhancements, and maintenance.

Impact Assessment:

1. **Safer Online Environment:** Contributed to fostering a safer and more respectful online community by reducing exposure to hate speech.
2. **User Empowerment:** Offered users greater control over their browsing experiences by allowing customization and settings adjustments.
3. **Educational Value:** Raised awareness regarding hate speech prevention and the importance of respectful online interactions.

Future Considerations:

1. **Continuous Enhancement:** Commitment to ongoing improvements based on user feedback and evolving language patterns.
2. **Regulatory Compliance:** Continued adherence to data privacy regulations and standards.
3. **User Education:** Focus on educating users about features, privacy practices, and the importance of responsible online behaviour.

Overall Conclusion:

The Hate Speech and Custom Word Blocking Extension has successfully addressed the initial objectives of providing users with a tool to mitigate exposure to hate speech content while offering customization options for a more tailored browsing experience. The project's conclusion marks a significant milestone, emphasizing the importance of fostering a respectful online environment and committing to ongoing improvements to meet evolving user needs and industry standards.

References

1. https://www.geeksforgeeks.com
2. https://www.wikipedia.com