Glimpse

Final Report

Group 2

Team members' details:

1. **Raman Sharma – 22114076**
   Contact No: +91 75218 29455
   Email ID: raman_s@cs.iitr.ac.in
   Contribution: Served as **Team leader**, contributed to **designing the whole UI prototype** and implementing the **complete bookmark feature**, including **custom bookmark tagging** on the front end.

2. **Boda Yashwanth – 22114022**
   Contact No: +91 89779 25111
   Email ID: boda_y@cs.iitr.ac.in
   Contribution: Served as **front-end developer** and **technical writer**. Responsible for developing the **prototype of the NewTab page** and implementing the **basic settings feature in the NewTab page**. Also, responsible for **writing all documentation** of the project.

3. **Ayush Ranjan – 22114018**
   Contact No: +91 74829 58551
   Email ID: ayush_r@cs.iitr.ac.in
   Contribution: Served as **Back-End developer**. Developed the **scrapper tools** and **text processors** powering **all our ML models**. Also contributed to the **speed, accuracy and reliability** testing of our ML models.

4. **Souvik Karmakar – 22114096**
   Contact No: +91 86429 24659
   Email ID: souvik_k@cs.iitr.ac.in
   Contribution: Served as **leading developer**, contributed to **core settings feature** and **Miniature Preview of Web Links** along with **estimated reading time** and **system integration.**

5. **Sarvasva Gupta – 22114086**
   Contact No: +91 79899 04811
   Email ID: sarvasva_g@cs.iitr.ac.in
   Contribution: Served as a **backend developer**, contributed to **Text Summarization, choosing model, and creating ML Tag Assignment to bookmark links, estimating the reading time** along with **creating required design diagram** relevant to the project. Also contributed to the testing of various ML models.

6. **Anvit Gupta – 22114009**
   Contact No: +91 94620 11044
   Email ID: anvit_g@cs.iitr.ac.in
   Contribution: Served as **Module and System tester**. Contributed to **intermediate documentations.**

7. **Vineet Kumar – 22114107**
   Contact No: +91 92634 36403
   Email ID: vineet_k@cs.iitr.ac.in
   Contribution: Served as **Front-End Developer** and **Technical Writer**. I contributed to the **prototype of the new tab page** and made the **Greetings** and **Motivational Quotes** and **To-Do Lists**. Also helped with various Documents.

Introduction

Glimpse aims to enhance users' browsing experience by providing an array of productivity features that are seamlessly integrated into their browsing environment. The extension facilitates efficient bookmarking with automatic tag assignment using machine learning algorithms, offers customisable tagging options, enables organised To - Do lists with tag-based organisation, displays motivational quotes on the browser's homepage, and introduces a unique miniature preview feature for web links.

The important features of the Glimpse are explained in short as follows:

1. **Miniature Preview of Web Links:** A standout feature of the extension is its miniature preview functionality, which offers users a glimpse into the content of web links without having to navigate away from their current page. By hovering over a link, users can view a scrollable preview of the linked page's content, along with a summary generated through machine learning analysis. This feature aids in quick decision-making regarding the relevance of the linked content, thereby optimizing browsing efficiency.

2. **Bookmarking with ML Tag Assignment:** The extension leverages machine learning algorithms to automatically assign relevant tags to bookmarked web pages. This feature eliminates the manual effort required for tagging and improves organisation efficiency.

3. **Custom Tag Feature:** Users can create personalised tags to further organise their bookmarks and Todo lists according to their preferences. This feature provides flexibility and customisation options tailored to individual user needs.

4. **To-Do Lists Organization:** To-do lists within the extension are organised using tags, allowing users to categorise tasks efficiently. By grouping tasks based on specific criteria, users can manage their tasks more effectively and prioritise their workflow.

5. **Motivational Quotes on Home Page:** The extension injects motivation into users' browsing sessions by displaying inspirational quotes on the browser's homepage. These quotes serve as uplifting reminders to stay focused and positive throughout their online activities.

With its diverse set of features geared towards productivity and user convenience, our browser extension aims to streamline the browsing experience for users. By combining machine learning capabilities with customizable organization tools and motivational elements, the extension empowers users to make the most of their online activities while staying focused and inspired.

Motivation for the Project

The ever-growing volume of information available online presents challenges in efficiently navigating and extracting valuable content. This project was born from a desire to address these challenges and enhance the overall browsing experience. We envisioned a user-friendly Chrome extension that would streamline information retrieval, promote focus, and ultimately make web browsing a more productive and enjoyable activity.

Application Domain

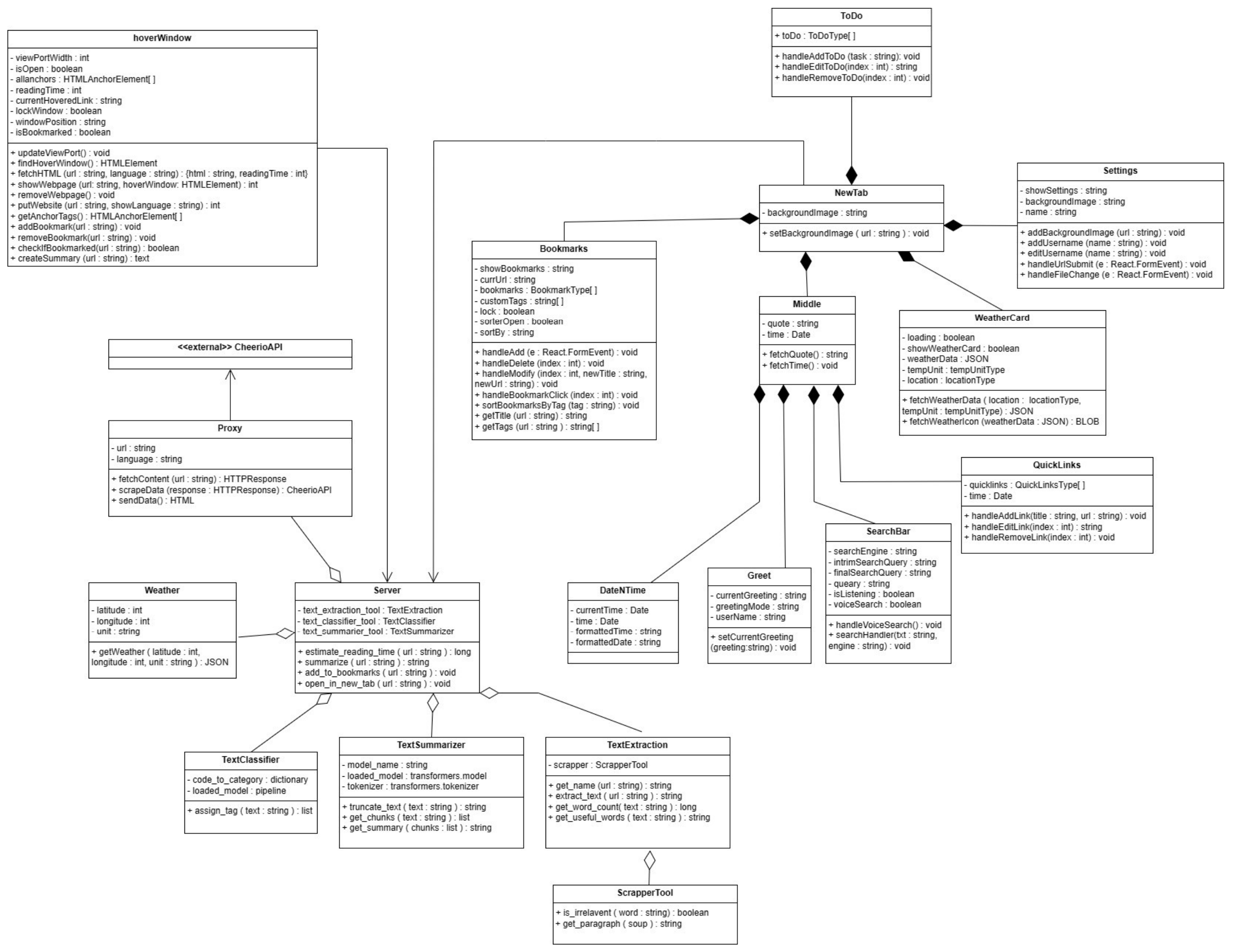
Glimpse is a Chrome extension designed to function as both a productivity tool and an efficient information management tool.

- **Productivity Tool:** It empowers users to streamline their browsing experience by offering features like:
    - ➢ Customizable new tab interface with to-do list management for prioritizing tasks.
    - ➢ Website previews with estimated reading time for quicker information assessment.
    - ➢ Distraction-free UI to minimize browsing interruptions and enhance focus.
- **Efficient Information Management Tool:** Glimpse equips users with functionalities to effectively navigate and organize online information:
    - ➢ Bookmark management system for saving and accessing important webpages.
    - ➢ Tools for summarising the content of website without even opening it.

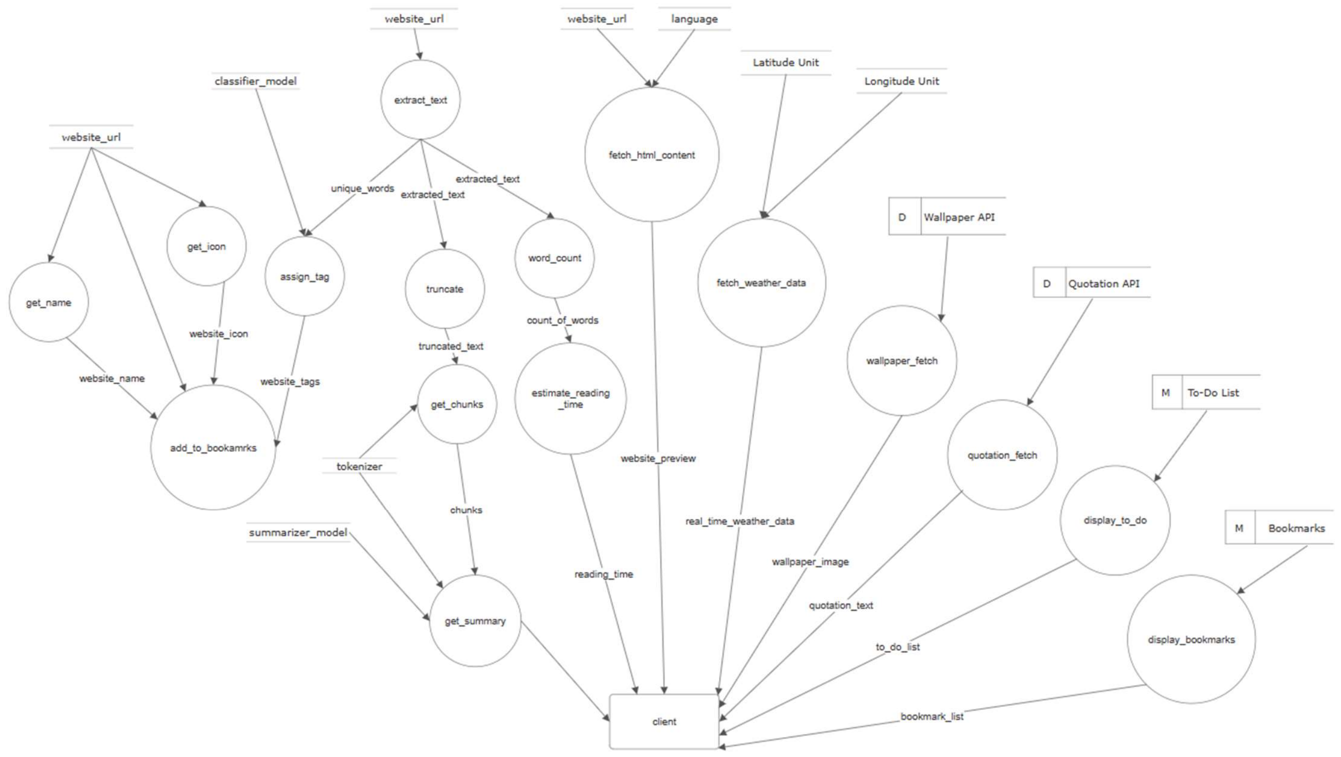
By combining features that enhance productivity and information management, Glimpse aims to make web browsing a more efficient and enriching experience for its users.

Design of the System

Class Diagram:


ToDo
+ toDo : ToDoType[ ]
+ handleAddToDo (task : string): void
+ handleEditToDo(index : int) : string
+ handleRemoveToDo(index : int) : void
hoverWindow
- viewPortWidth : int
- isOpen : boolean
- allanchors : HTMLAnchorElement[ ]
- readingTime : int
- currentHoveredLink : string
- lockWindow : boolean
- windowPosition : string
- isBookmarked : boolean
+ updateViewPort() : void
+ findHoverWindow() : HTMLElement
+ fetchHTML (url : string, language : string) : {html : string, readingTime : int}
+ showWebpage (url: string, hoverWindow: HTMLElement) : int
+ removeWebpage() : void
+ putWebsite (url : string, showLanguage : string) : int
+ getAnchorTags() : HTMLAnchorElement[ ]
+ addBookmark(url : string) : void
+ removeBookmark(url : string) : void
+ checkIfBookmarked(url : string) : boolean
+ createSummary (url : string) : text
Settings
- showSettings : string
- backgroundImage : string
- name : string
+ addBackgroundImage (url : string) : void
+ addUsername (name : string) : void
+ editUsername (name : string) : void
+ handleUrlSubmit (e : React.FormEvent) : void
+ handleFileChange (e : React.FormEvent) : void
NewTab
- backgroundImage : string
+ setBackgroundImage (url : string) : void
Bookmarks
- showBookmarks : string
- currUrl : string
- bookmarks : BookmarkType[ ]
- customTags : string[ ]
- lock : boolean
- sorterOpen : boolean
- sortBy : string
+ handleAdd (e : React.FormEvent) : void
+ handleDelete (index : int) : void
+ handleModify (index : int, newTitle : string, newUrl : string) : void
+ handleBookmarkClick (index : int) : void
+ sortBookmarksByTag (tag : string) : void
+ getTitle (url : string) : string
+ getTags (url : string) : string[ ]
Middle
- quote : string
- time : Date
+ fetchQuote() : string
+ fetchTime() : void
WeatherCard
- loading : boolean
- showWeatherCard : boolean
- weatherData : JSON
- tempUnit : tempUnitType
- location : locationType
+ fetchWeatherData (location : locationType, tempUnit : tempUnitType) : JSON
+ fetchWeatherIcon (weatherData : JSON) : BLOB
<<external>> CheerioAPI
Proxy
- url : string
- language : string
+ fetchContent (url : string) : HTTPResponse
+ scrapeData (response : HTTPResponse) : CheerioAPI
+ sendData() : HTML
QuickLinks
- quicklinks : QuickLinksType[ ]
- time : Date
+ handleAddLink(title : string, url : string) : void
+ handleEditLink(index : int) : string
+ handleRemoveLink(index : int) : void
SearchBar
- searchEngine : string
- intrimSearchQuery : string
- finalSearchQuery : string
- queary : string
- isListening : boolean
- voiceSearch : boolean
+ handleVoiceSearch() : void
+ searchHandler(txt : string, engine : string) : void
Greet
- currentGreeting : string
- greetingMode : string
- userName : string
+ setCurrentGreeting (greeting:string) : void
DateNTime
- currentTime : Date
- time : Date
- formattedTime : string
- formattedDate : string
Weather
- latitude : int
- longitude : int
- unit : string
+ getWeather (latitude : int, longitude : int, unit : string) : JSON
Server
- text_extraction_tool : TextExtraction
- text_classifier_tool : TextClassifier
- text_summarier_tool : TextSummarizer
+ estimate_reading_time (url : string) : long
+ summarize (url : string) : string
+ add_to_bookmarks (url : string) : void
+ open_in_new_tab (url : string) : void
TextClassifier
- code_to_category : dictionary
- loaded_model : pipeline
+ assign_tag (text : string) : list
TextSummarizer
- model_name : string
- loaded_model : transformers.model
- tokenizer : transformers.tokenizer
+ truncate_text (text : string) : string
+ get_chunks (text : string) : list
+ get_summary (chunks : list) : string
TextExtraction
- scrapper : ScrapperTool
+ get_name (url : string) : string
+ extract_text (url : string) : string
+ get_word_count(text : string) : long
+ get_useful_words (text : string) : string
ScrapperTool
+ is_irrelavent (word : string) : boolean
+ get_paragraph (soup) : string

Data Flow Diagram:



Implementation Detail of the System

Text Classifier Module:

- This module automatically categorizes text content. It uses a pre-trained machine learning model based on supervised learning techniques. The model likely analyzes the text using a method called TF-IDF (Term Frequency-Inverse Document Frequency) to understand the importance of words within the text. Based on this analysis, it classifies the text into predefined categories using a Multinomial Naive Bayes classifier.
- The module also employs a confidence threshold. For a category to be assigned, its probability score must be the highest (greater than or equal to 30%) and significantly higher than the second-highest probability (at least 20% of the highest score). This ensures a certain level of confidence in the assigned category.

Text Summarizer Module:

- This module creates summaries of longer pieces of text. It utilizes a pre-trained model from the Hugging Face community, specifically "facebook/large-bart-cnn". This model likely employs a technique called BART (Bidirectional and Attentive Transformer) to understand the relationships between words in the text.

- The module first prepares the text for processing by breaking it down into smaller chunks using a tokenizer. This ensures that each chunk stays within the model's capacity for handling information at once (typically limited to 1024 tokens).
- To improve efficiency, the module might also truncate the original text if it's excessively long. Finally, the chunked text is fed into the pre-trained model, which generates a concise summary of the original content.

Text Extraction Module:

This code module focuses on extracting and processing website content. It first filters out irrelevant sections of a webpage, like headers and footers, using a predefined list of HTML tags. Then, it specifically targets text within paragraph elements to gather the main body content. The extracted text is then cleaned up using a spaCy library. This cleaning process removes common words (like "the" or "a"), punctuation, and non-alphanumeric characters, essentially leaving behind the core textual information. Finally, the module extracts the domain name from the URL and calculates the total word count of the cleaned website text. Overall, this module acts as a preparation step, transforming raw website content into usable data for further analysis or tasks.

New Tab Module:

The New Tab module provides a customizable and informative experience for your browser's new tab page. Here's a breakdown of its key components:

1. Bookmarks Module:

- Manages your saved bookmarks, including adding, deleting, and editing them.
- Uses the Text Classifier Module to automatically suggest tags for bookmarked websites.
- Allows sorting bookmarks by different criteria.

2. Middle Module:

- Displays the current date and time.
- Provides a motivational quote with username (potentially fetched from an external source).
- Includes a search bar with engine selection and voice search option (if applicable).
- Offers quick links for easy access to frequently visited websites (with management functionalities).

3. Weather Card Module:

- Shows the weather information for your location, retrieved using the OpenWeatherMap API.

4. Settings Module:

- Allows customization of the background image for the new tab page.
- Enables adding and editing a username for personalized greetings.

5. To Do List Module:

- Manages your to-do list within the new tab, allowing task creation, editing, and deletion.

This comprehensive New Tab module provides a central hub for essential information, quick access points, organization tools, and a touch of personalization for a more efficient and engaging browsing experience.

Backend Server:

1. **Express Application:** The main entry point of the application. It sets up the server and listens for incoming requests.
2. **Middleware:** The application uses two middleware functions: express.json() for parsing incoming request bodies, and cors() for handling Cross-Origin Resource Sharing (CORS).
3. **Cache:** The application uses a simple in-memory cache to store weather data. The cache is loaded from a file when the application starts, and it's saved to a file whenever new data is added to the cache.

4. **API Endpoints:** The application provides several API endpoints:

   - /api/v1/weather: This endpoint accepts a POST request with latitude, longitude, and units query parameters. It returns weather data for the specified location. The data is fetched from the OpenWeatherMap API and cached for 1 hour.

   - /api/v1/urldata: This endpoint accepts a POST request with a URL in the request body. It fetches the HTML of the specified website, parses it using an external library Cheerio, and returns the title, description, keywords, and website author.

   - /api/v1/icon: This endpoint accepts a POST request with a URL in the request body. It fetches the HTML of the specified website, parses it using Cheerio, and returns the URL of the website's favicon.

   - /api/v1/proxy: This endpoint accepts a POST request with a “url” and “showLanguage” in the request body. It fetches the HTML of the specified website in the specified language that the user requested using Axios (external library), rewrites the URLs of linked resources to be absolute URLs, calculates the estimated reading time of the page from the word count (200 words/min), and returns the modified HTML as a response to and the estimated reading time as a response header (x-reading-time).

- /api/v1/summary: This endpoint accepts a POST request with a URL of a website in the request body and creates a summary of the site content using the “Server Module” and sends the summary as a text file.

- /api/v1/get_tags: This endpoint accepts a POST request with a website URL in the request body and generates appropriate tags according to the page's content using the “Server Module”. The response sends the tags list generated.

5. **Error Handling:** The application has basic error handling. If an error occurs while processing a request, it logs the error and returns the appropriate error along with a suitable HTTP status code.

6. **Environment Variables:** The application uses the dotenv package to load environment variables from a .env file. The OpenWeatherMap API key is loaded from the WEATHER_API_KEY environment variable.

Tools and Technologies Used in the System

Front-End

- **TypeScript -** TypeScript shines in several areas for JavaScript development. Its static typing catches errors early, improving code quality. Rich IDE support makes working with large codebases easier. Features like interfaces and generics enhance code organisation and reusability. Overall, TypeScript offers a more robust and collaborative development experience, making it a great choice for complex projects.

- **Cascading Style Sheets -** CSS empowers developers to create visually appealing websites with a clear separation of design (CSS) from content (HTML), leading to cleaner, more maintainable code and a consistent user experience across different devices.

- **Web Storage API** - Local storage offers convenient data storage on users' devices, bypassing complex user account management. It's ideal for website preferences, temporary app data, and caching frequently accessed information for a faster, more personalised user experience.

- **Weather API** – Used OpenWeather API for getting real-time weather details

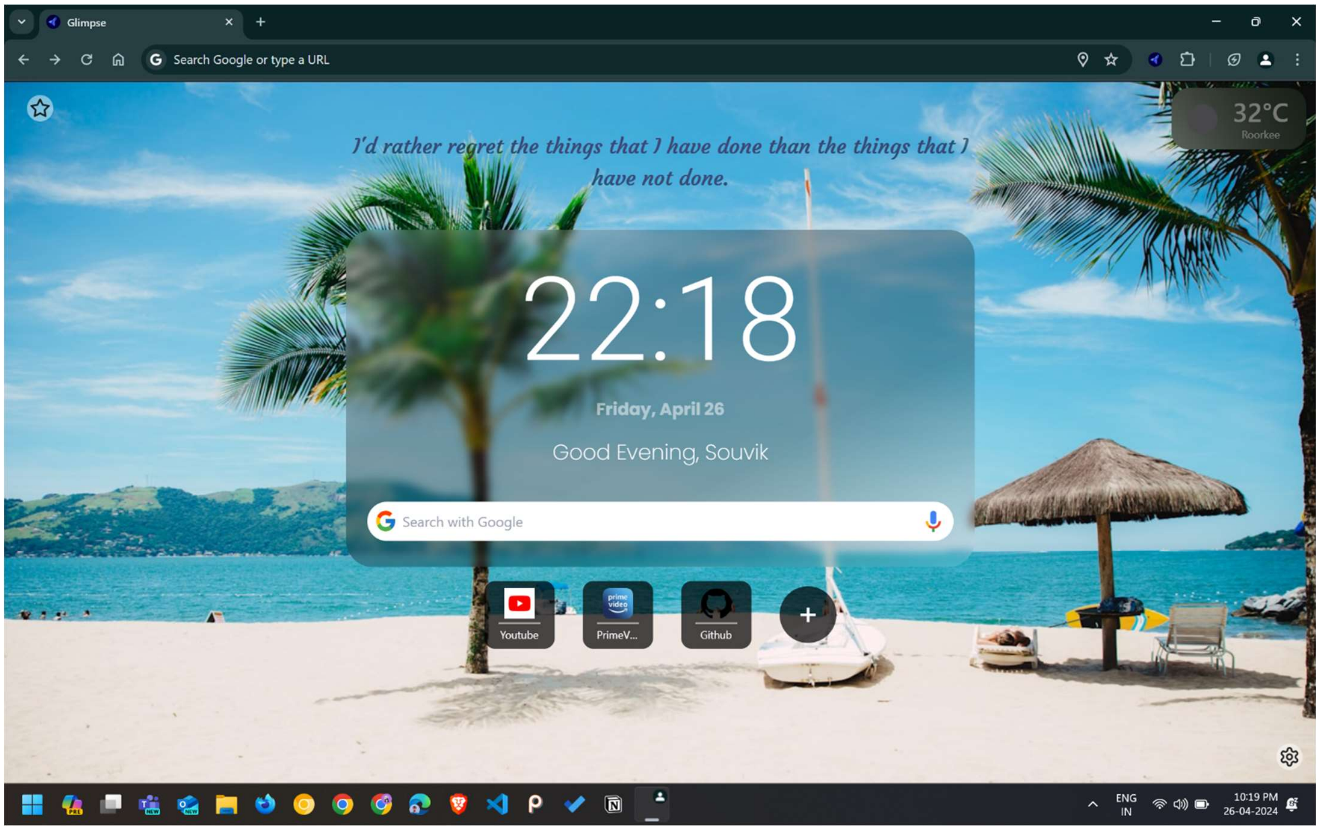
- **Background Image API** – It’s used to fetch background given labels or keywords

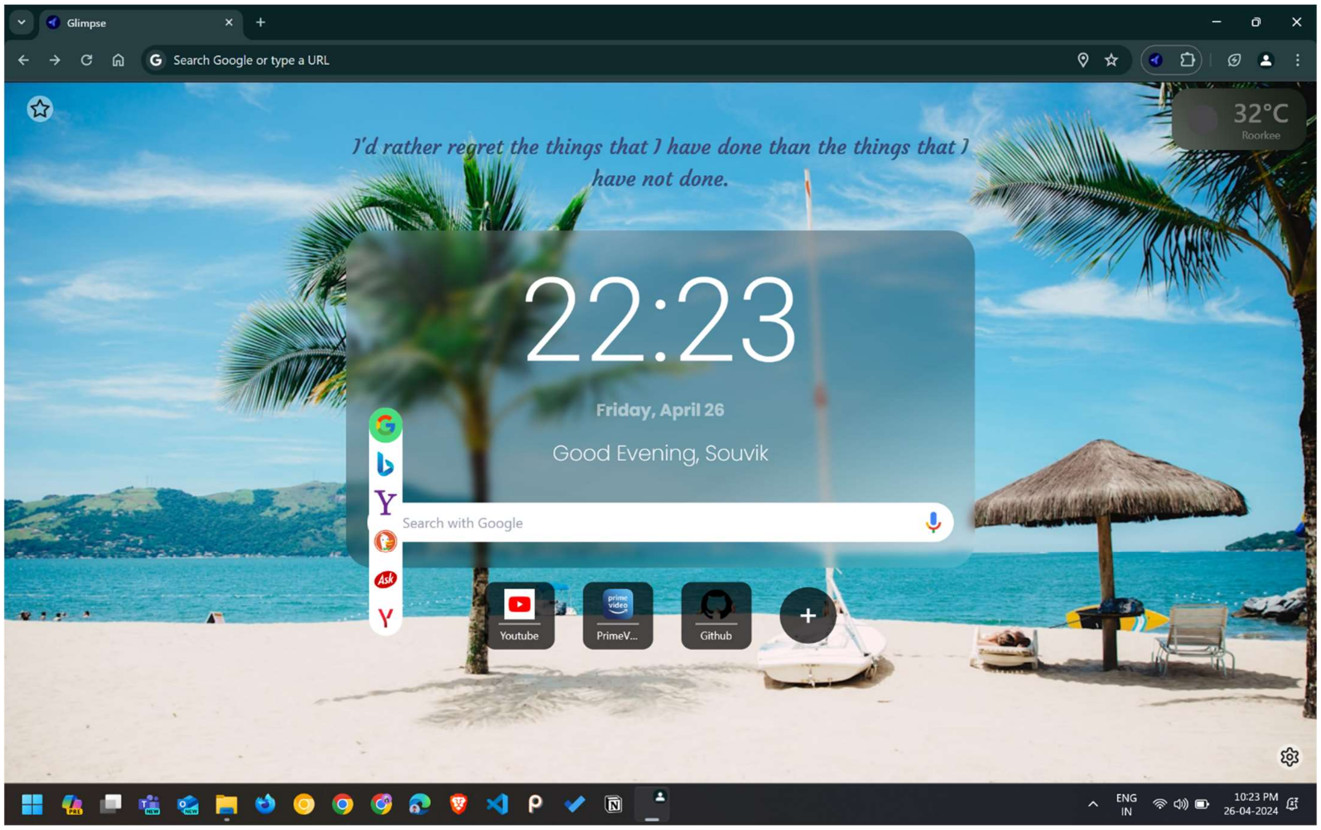
- **Quotes API** – Used to Quotable.io to display a motivational quote each time the user opens new tab.

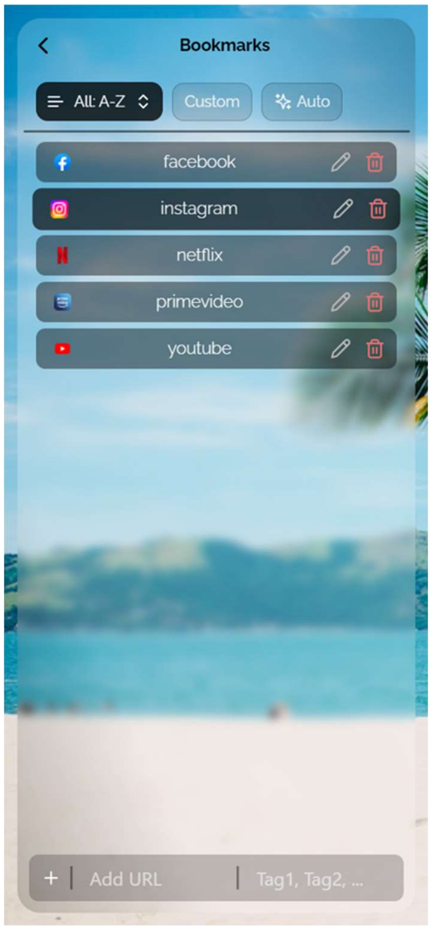
Back-End

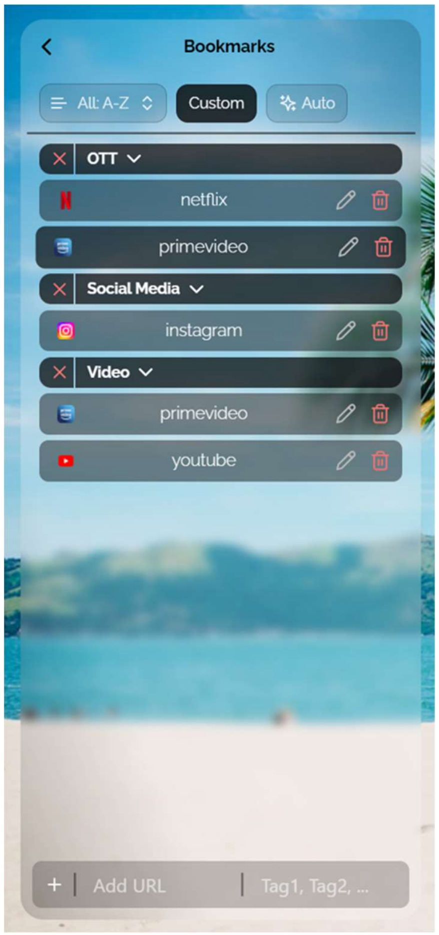
- An ML model based on Supervised Learning for Text Classification utilising a TF-IDF Vectorizer and Multinomial Naïve Bayes Classifier.
- Text Summarizer Model - Pre-trained model from the Hugging-Face community – "facebook/large-bart-cnn"
- Python Libraries
    - BeautifulSoup: Used to parse HTML and get website content
    - spaCy: Used to extract useful words from the website text

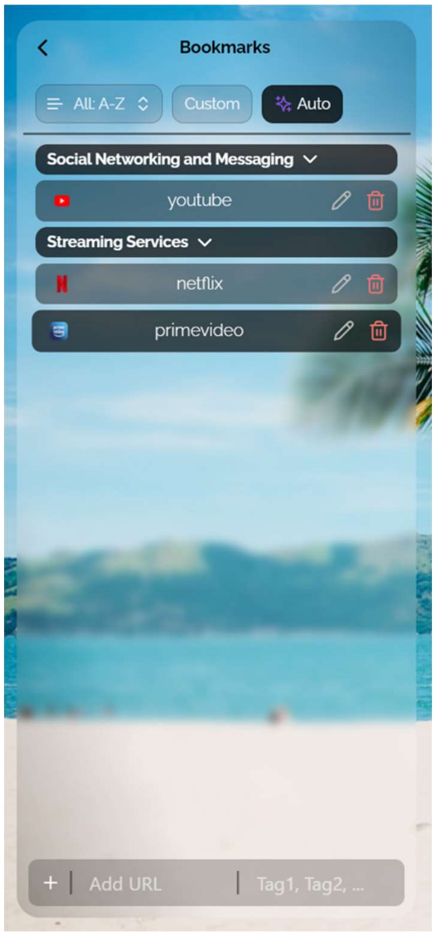
Screenshots of the System

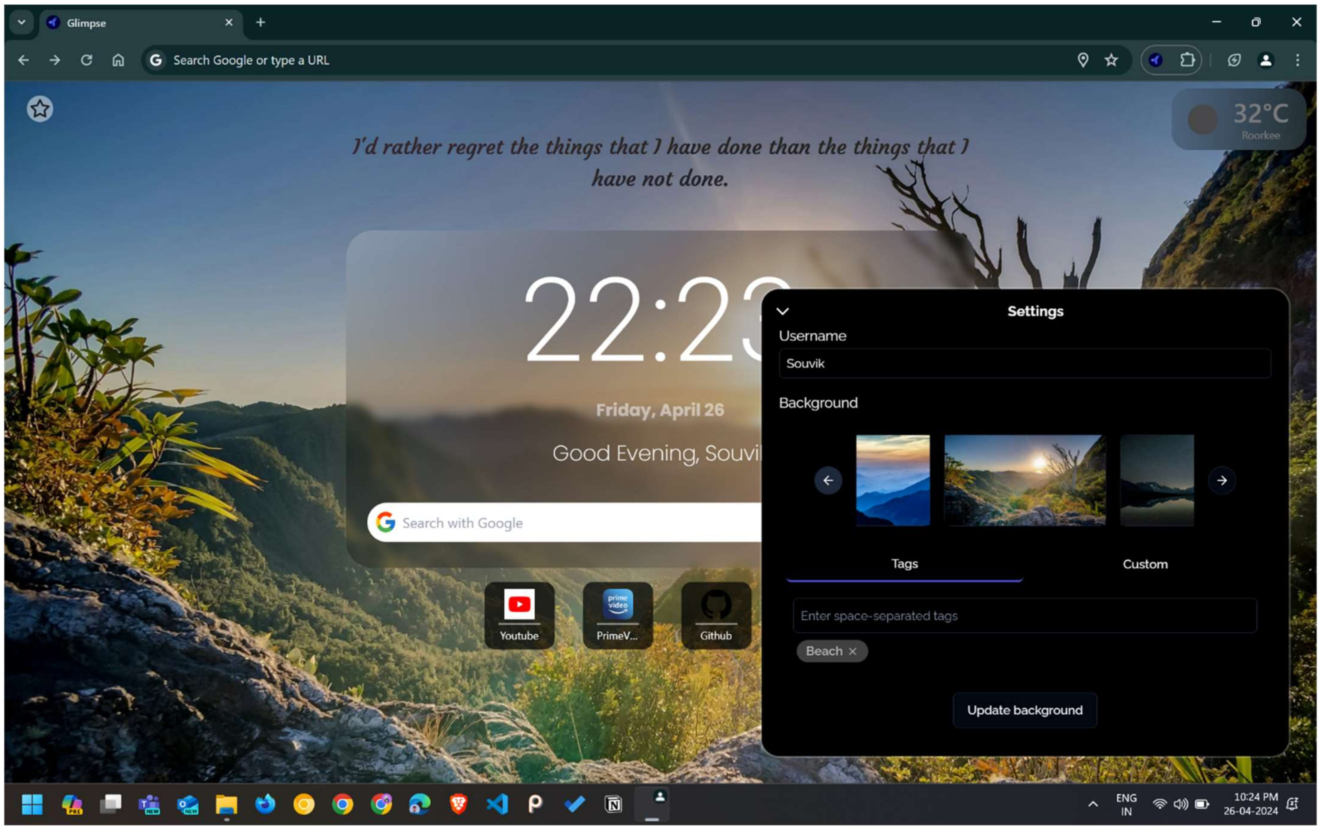



Glimpse
Search Google or type a URL
32°C
Roorkee
I'd rather regret the things that I have done than the things that I have not done.
22:23
Friday, April 26
Good Evening, Souvik
Search with Google
Youtube
PrimeV...
Github
ENG
IN
10:23 PM
26-04-2024

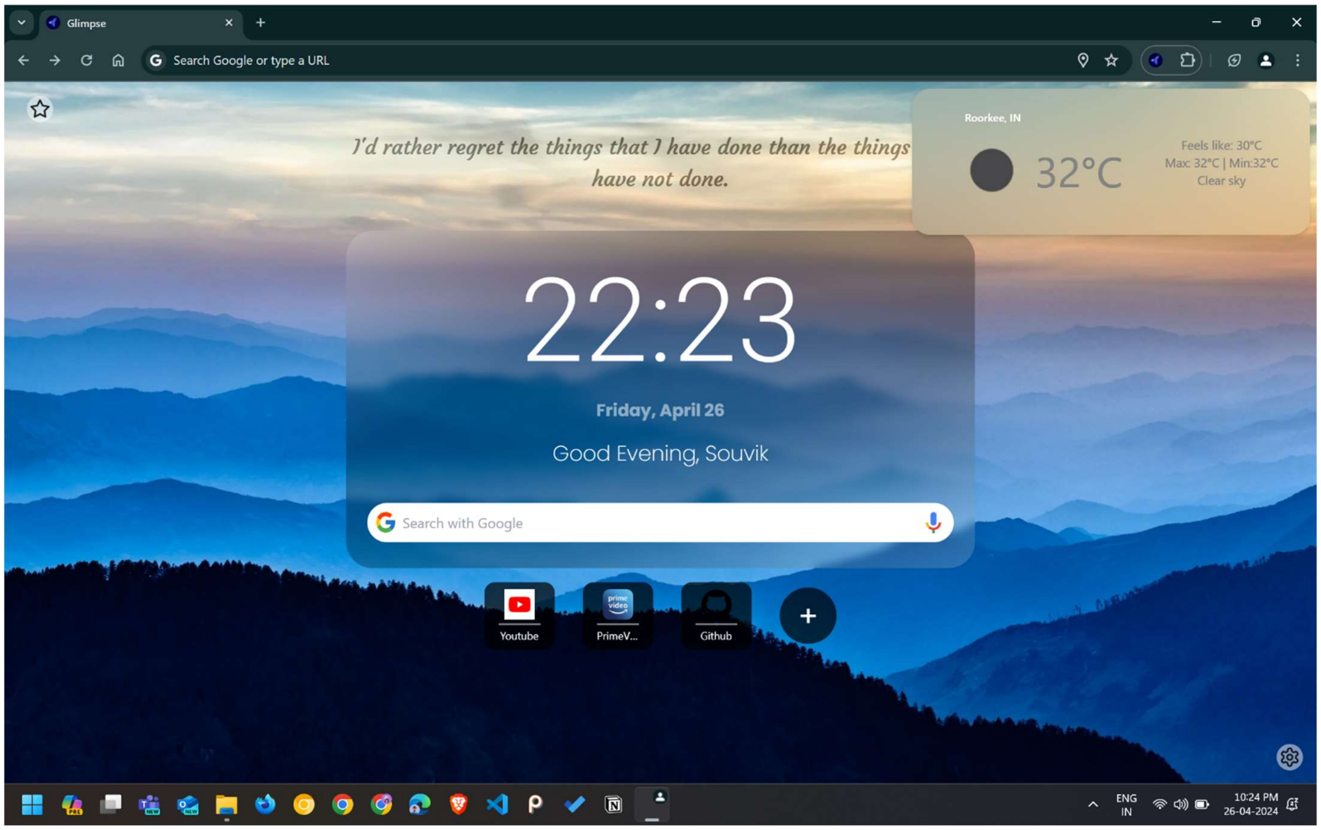

Bookmarks
All: A-Z
Custom
Auto
facebook
instagram
netflix
primevideo
youtube
Add URL
Tag1, Tag2, ...


Bookmarks
All: A-Z
Custom
Auto
OTT
netflix
primevideo
Social Media
instagram
Video
primevideo
youtube
Add URL
Tag1, Tag2, ...

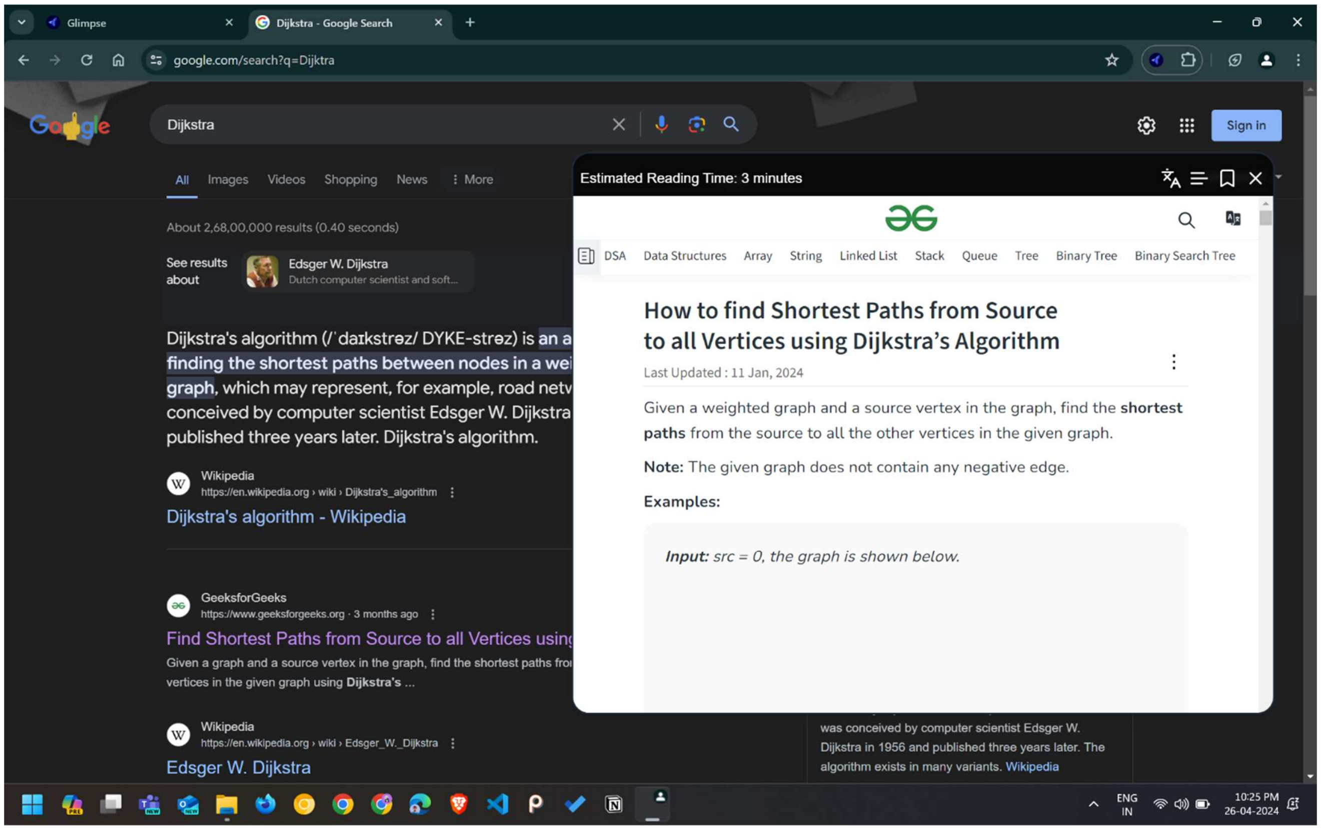

Bookmarks
All: A-Z
Custom
Auto
Social Networking and Messaging
youtube
Streaming Services
netflix
primevideo
Add URL
Tag1, Tag2, ...



Glimpse
Search Google or type a URL
32°C
Roorkee
I'd rather regret the things that I have done than the things that I have not done.
22:23
Friday, April 26
Good Evening, Souvi
Search with Google
Youtube
PrimeV...
Github
Settings
Username
Souvik
Background
Tags
Custom
Enter space-separated tags
Beach
Update background



Glimpse
Search Google or type a URL
Roorkee, IN
32°C
Feels like: 30°C
Max: 32°C | Min:32°C
Clear sky
I'd rather regret the things that I have done than the things
have not done.
22:23
Friday, April 26
Good Evening, Souvik
Search with Google
Youtube
PrimeV...
Github



About 2,68,00,000 results (0.40 seconds)

See results about Edsger W. Dijkstra — Dutch computer scientist and soft...

Dijkstra's algorithm (/ˈdaɪkstrəz/ DYKE-strəz) is an a finding the shortest paths between nodes in a wei graph, which may represent, for example, road netv conceived by computer scientist Edsger W. Dijkstra published three years later. Dijkstra's algorithm.

Wikipedia
https://en.wikipedia.org › wiki › Dijkstra's_algorithm
Dijkstra's algorithm - Wikipedia

GeeksforGeeks
https://www.geeksforgeeks.org · 3 months ago
Find Shortest Paths from Source to all Vertices using
Given a graph and a source vertex in the graph, find the shortest paths fro vertices in the given graph using **Dijkstra's** ...

Wikipedia
https://en.wikipedia.org › wiki › Edsger_W._Dijkstra
Edsger W. Dijkstra

was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later. The algorithm exists in many variants. Wikipedia

Estimated Reading Time: 3 minutes

How to find Shortest Paths from Source to all Vertices using Dijkstra's Algorithm

Last Updated : 11 Jan, 2024

Given a weighted graph and a source vertex in the graph, find the **shortest paths** from the source to all the other vertices in the given graph.

Note: The given graph does not contain any negative edge.

Examples:

Input: *src = 0, the graph is shown below.*

Comparison with Existing Systems

Features supported	Glimpse	One tab	Pocket	Raindrop.io
Bookmark	Yes	No	Yes	Yes
Website Preview	Yes	No	No	No
Search bar	Yes	No	No	No
ML Integration	Yes	No	No	No
Focused Interface	Yes	No	Yes	No
Tab Grouping	No	Yes	No	No

Following are the systems which provide similar features as our project:-

1. **OneTab:**

Strengths:

a. OneTab consolidates open tabs into a single list, improving browser performance.

Weaknesses:

a. Limited bookmark management and tagging capabilities.

b. Lacks a built-in To-do list feature.

2. **Pocket:**

Strengths:

a. Enables users to save articles, videos, and web pages for later viewing.

b. Provides a clean and distraction-free reading environment.

Weaknesses:

a. It focuses on saving content rather than managing a comprehensive bookmark and Todo list.

b. Limited support for website previews.

3. **Raindrop.io:**

Strengths:

a. Robust bookmark management with folder organization and tagging.

b. Supports collaborative bookmarking for teams.

Weaknesses:

a. Limited Todo list integration.

b. Website previews may not be a primary focus.

Outcome of the Project

Project Deliverables

- **Functional Product Development**: Completion of a functioning Chrome extension that provides users with multiple productivity tools for a fast and focused browsing experience.

- **New Tab Feature:** Successful implementation of a clutter-free interface designed for focus. Access your essential Bookmarks instantly, maximising your productivity when you open a new tab.

- **Backend Component Integration:** Effective integration and functioning of Text classification, Text summarizer and Preview components.

Achieved Objectives

- **We achieved enhanced efficiency** by enabling users to quickly preview website content. This allows them to evaluate relevance before opening full pages, saving valuable time for research and general browsing.
- **We achieved streamlined organization** by centralizing bookmark management, to-do lists, and personalized quotes. This promotes focus and reduces information overload by providing a single location for essential information.
- **We achieved improved focus** by offering a simplified view of all features in the new tab. This view removes distracting elements and allows users to concentrate solely on the core content of webpages.
- **We achieved deeper understanding** by incorporating AI/ML-powered features like content summaries. This facilitates a quicker grasp of complex information, enhancing overall comprehension.

Challenges Faced

- We faced a challenge in choosing which storage option to use between Local Storage API or Database Storage. At the end, we decided to use the Local Storage API.

- We faced a challenge while trying to access a website directly from our browser. The Cross-Origin Resource Sharing (CORS) policy of the server hosting the website blocks access to it. This policy allows only specific IP addresses and search engine like Google or Microsoft Edge, to access the webpage. To bypass this, we decide to use a backend proxy server to fetch the website content and show it to the user.

Lessons Learnt

The Glimpse project provided invaluable lessons in team collaboration and communication. Emphasising clear communication channels and regular updates, the team learned the significance of transparency and understanding among members.

Utilising collaborative tools efficiently, delegating tasks clearly, and fostering an open environment for idea-sharing and feedback promoted innovation and collective ownership. Addressing conflicts promptly and constructively, along with encouraging constructive feedback, facilitated a harmonious working environment.

These insights are a foundation for improved teamwork, enhancing efficiency and communication in future projects.

Bibliography

These are some of the resources which we used while developing this project:

1. https://developer.chrome.com/docs/extensions
2. https://docs.plasmo.com
3. https://youtu.be/SqcY0GlETPk?si=4PAu9v0Jsv5INMB2
4. https://docs.docker.com/build/cloud/?_gl=1*13vgobr*_ga*MjAxNjMzNTgwLjE3MTQxNDE4NjU.*_ga_XJWPQMJYHQ*MTcxNDE0MTg2NS4xLjEuMTcxNDE0MTg2Ni41OS4wLjA.
5. https://www.kaggle.com/datasets
6. https://huggingface.co/models
7. https://www.nltk.org/
8. https://spacy.io/