Glimpse

Design Document

Date: 26th March 2024
Version: 1.0

Group – 2

1. **Raman Sharma – 22114076**
Contact No: +91 75218 29455
Email ID: raman_s@cs.iitr.ac.in

2. **Boda Yashwanth – 22114022**
Contact No: +91 89779 25111
Email ID: boda_y@cs.iitr.ac.in

3. **Ayush Ranjan – 22114018**
Contact No: +91 74829 58551
Email ID: ayush_r@cs.iitr.ac.in

4. **Souvik Karmakar – 22114096**
Contact No: +91 86429 24659
Email ID: souvik_k@cs.iitr.ac.in

5. **Sarvasva Gupta – 22114086**
Contact No: +91 79899 04811
Email ID: sarvasva_g@cs.iitr.ac.in

6. **Anvit Gupta – 22114009**
Contact No: +91 94620 11044
Email ID: anvit_g@cs.iitr.ac.in

7. **Vineet Kumar – 22114107**
Contact No: +91 92634 36403
Email ID: vineet_k@cs.iitr.ac.in

Summary of Design

Glimpse aims to enhance users' browsing experience by providing an array of productivity features seamlessly integrated into their browsing environment. The extension facilitates efficient bookmarking with automatic tag assignment using machine learning algorithms, offers customizable tagging options, enables organized To - Do lists with tag-based organization, 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 organization efficiency.

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

4. To - Do Lists Organization: Todo lists within the extension are organized using tags, allowing users to categorize tasks efficiently. By grouping tasks based on specific criteria, users can manage their tasks more effectively and prioritize their workflow.

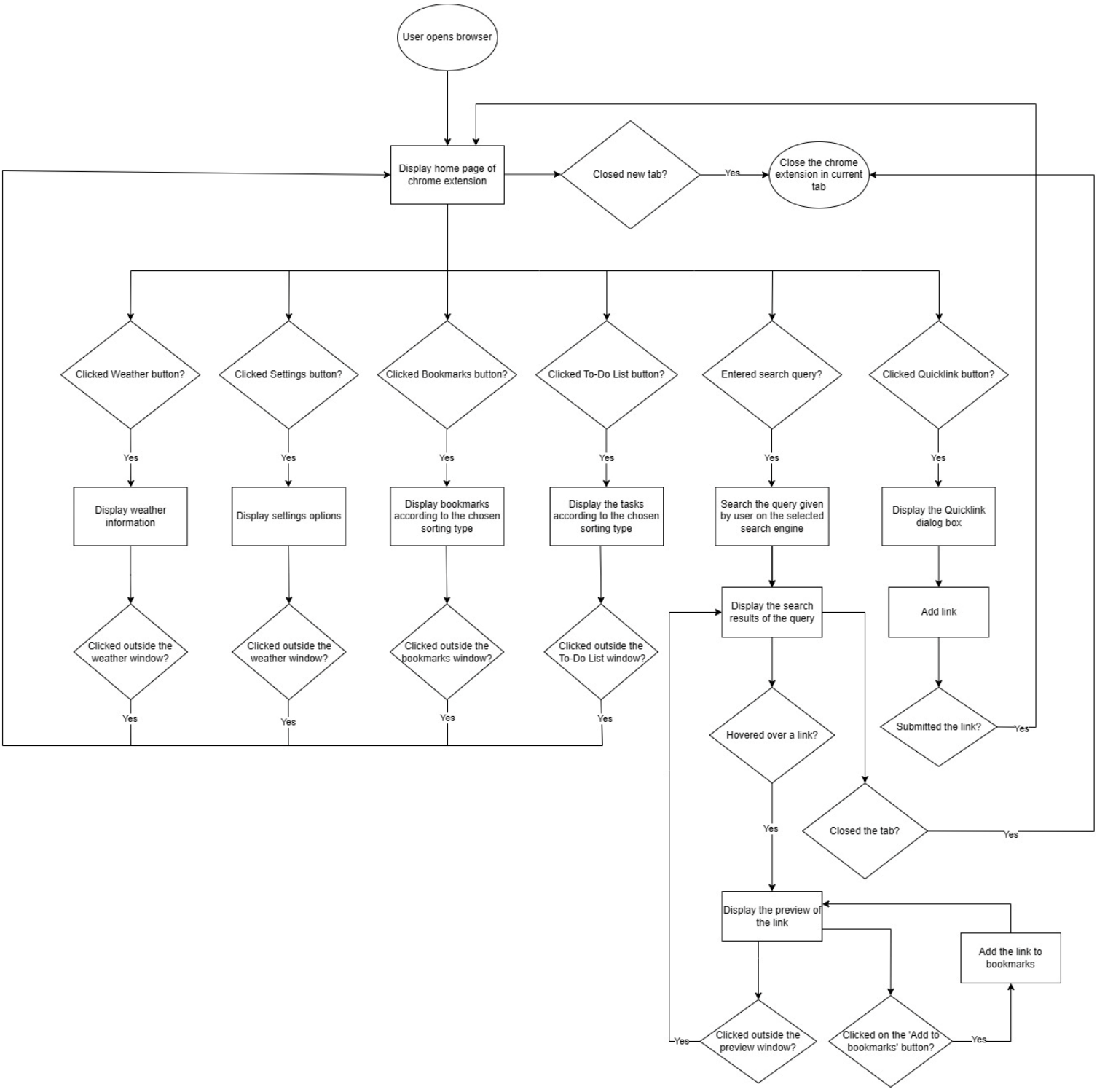
5. Motivational Quotes on Home Page: The extension injects motivation into users' browsing sessions by displaying inspirational quotes on the browser's homepage. These quotes serve as uplifting reminders to stay focused and positive throughout their online activities.

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.

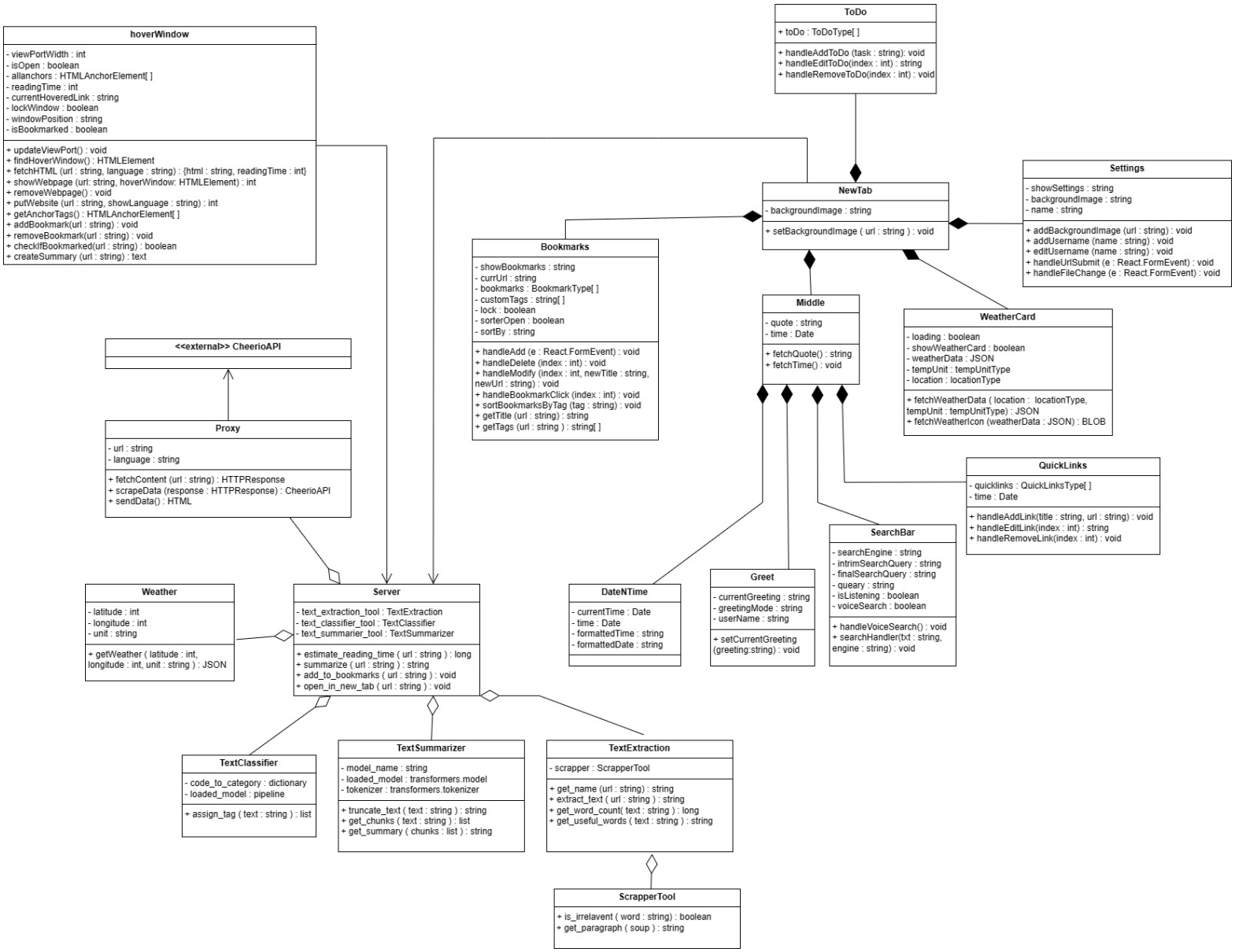
Main Design

High Level Design:

- **Flowchart**



- **Class Diagram**



- **Data Flow Diagram**

Low Level Design:

Text Classifier Module:

Data Items:

- code_to_category: A dictionary for converting the code of website categories obtained into text form.
- loaded_model: An ML model based on Supervised Learning for Text Classification utilizing a TF-IDF Vectorizer and Multinomial Naïve Bayes Classifier.

Functions:

- assign_tag(text): Assigns tags to the input text based on the following criteria: Highest Probability >= 30% and Second Highest >= Minimum (20%, Highest Probability * 0.5).

Text Summarizer Module:

Data Items:

- model: Pre-trained model from the Hugging-Face community – "facebook/large-bart-cnn".
- tokenizer: The Tokenizer used for tokenization of text to prepare it for input to the specified model.

Functions:

- get_chunks(text): Breaks down the text into multiple token sequences so that the maximum number of tokens in each sequence doesn't exceed the maximum allowed by the tokenizer (typically, 1024 tokens).

- truncate_text(text): Truncates the input text to reduce the time required for summary generation.
- get_summary(chunks): Takes a list of chunks as input and generates a summary using the model.

Text Extraction Module:

Data Items:

- website_url: The URL of the website being processed.
- website_text: The extracted text content from a website.
- useful_word_lists: Extracted useful words from the website text.
- irrelevant_tags: a list of HTML tags to ignore while extracting text from the website.

Functions:

1. **ScrapperTool Class:**

- isIrrelevant(element): Determines if an HTML element belongs to the irrelevant tags category (e.g. header, footer) not useful for content extraction.
- get_paragraph(soup): Takes a soup object, extracts text contained in the paragraph HTML tag of the website.

2. **TextExtraction Class:**

- get_useful_words(text): Extracts useful words from the website text using spaCy library of Python, removing stopwords, punctuation, and non-alphanumeric characters.
- extract_text(url): Fetches website content, parses HTML using BeautifulSoup library of Python, utilizes get_paragraph() function of the ScrapperTool class to extract website text from the soup object, calls get_useful_words() as a subroutine to return a list of useful words.
- get_name(url): Extracts the domain name from a given URL.
- get_word_count(text): Returns the count of words in the text extracted from the website.

New Tab Module:

The 'New Tab' module has several sub-modules as mentioned below:-

1. **Bookmarks Module:**

Data Items:

- bookmarks[]: Store the bookmarks in a vector
- customTag[[]]: Stores the corresponding tag(s) of respective bookmarks
- sortBy: A string to ask how to sort the bookmarks

Functions:

- handleAdd(), handleDelete(bookmark), handleModify(bookmark): Handle adding, deleting and editing of bookmarks
- getTags(url): calls the Text Classifier Module for tags of given URL
- sortBookmarksBy(sortBy): Sort the bookmarks according to the chosen order

2. **Middle (DateTime, Greeting, Search Bar, QuickLink) Module:**

Data Items:

- formattedDate and formattedTime: will use fetchTime()
- currentGreeting: the quote along with the username
- query: the query of SearchBar
- engine: which search engine to use
- voiceSearch: if using voice search mode
- quickLinks[]: store all the quick links

Functions:

- searchHandler(query, engine): send the query to corresponding search engine
- handleAddLink(), handleEditLink(index), HandleRemoveLink(index): functions to add and edit quicklink

3. **Weather Card Module:**

Data Items:

- Location: User's location
- WeatherDetails: Details returned by the API call

Functions:

- fetchWeatherData(location): using OpenWeatherMap API to fetch data using the location as input

4. **Settings Module:**

Data Items:

- Backgroundimage: for custom background image
- UserName: name of user stored

Functions:

- addBackgroundImage(image): to add user custom image for background
- addUserName(), editUserName(): Interface to change and add Username for greetings

5. **To Do List Module:**

Data Items:

- ToDoItems(name, checkbox, deadline)[]: an array to store all To - Do items

Functions:

- handleAddToDo(todo-info), handleEditToDo(index), handleRemoveToDo(index): Handle adding, editing and deletion of tasks from To - Do list

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 author of the website.

    - /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 URL of a website in the request body and generates appropriate tags according to the content of the page 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 appropriate error along with 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.

Questions

1. **How will we gather a diverse dataset for training the machine learning model?**

Ans. We will use Datasets available on Kaggle for collecting a collection of words to tag information. As far as the Summarization module is concerned, we will use a pre-trained model from hugging-face community.

2. **Which machine learning algorithms are suitable for tag assignment and content summarization?**

Ans. As far as the assignment of tags is concerned, it is a type of Supervised Learning Algorithm. Summarization can be an Unsupervised Learning Algorithm.

3. **How can we design an intuitive interface for bookmarking and tagging web pages?**

Ans. We'll incorporate a simple yet functional UI with features like drag-and-drop tagging and one-click bookmarking.

4. **Which APIs or databases can we integrate for fetching motivational quotes?**

Ans. We'll explore popular quote APIs or curate a database of motivational quotes for seamless integration.

5. **How can we optimize the performance of the extension, especially when generating miniature previews?**

Ans. We'll implement caching mechanisms to reduce the need for repeated page loading and optimize resource usage.

6. **Why are we using a proxy server to show preview?**

Ans: If we try 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 edge, to access the webpage. To bypass this, we are using a backend server to fetch the website content and show it to the user.

7. **Why are we rewriting the URLs of the website HTML to be absolute URL?**

Ans: Almost each of the modern websites uses CSS (Cascading Style Sheet) and JavaScript. The browser will not properly render the page (How its meant to be) without all the resources (CSS and JavaScript) associated with the html. Generally, theses recourses' locations are written in relative format. The browser can fetch those resources with some sophisticated way, but in our case, we must pass those contents as absolute URL so that the browser can render the page properly.

8. What are the security risks to consider while creating a server-side proxy?

Ans:

- Man-in-the-Middle Attacks:

A server-side proxy sits between your webpage and the external website. This creates a potential vulnerability. A malicious actor could compromise the proxy server and tamper with the content being fetched. This could lead to data theft or malicious code injection.

- Server-Side Request Forgery (SSRF):

Improperly configured server-side proxies can be vulnerable to SSRF attacks. An attacker crafts a request that tricks the proxy into fetching data from unintended locations on your server's internal network. This could potentially expose sensitive information stored on your server, like databases or internal resources.

- Potential Violation of Terms of Service:

Some websites may have terms of service that prohibit scraping or bypassing their CORS restrictions. Using a proxy to access their content could violate these terms and potentially lead to your website being blocked.