Aymane Bokhamy

Senior Project Proposal

*QuickSummary: A Browser Extension for Article Summarization*

Section: Maria Webb

September 17, 2024

Table of Contents

[Project Summary 2](#_Toc175564289)

[Significance 3](#_Toc175564290)

[Required Tools & Availability 4](#_Toc175564291)

[Demonstration Plans 5](#_Toc175564292)

[Qualifications 5](#_Toc175564293)

[Project Specifications 6](#_Toc175564294)

[Functional Specifications 6](#_Toc175564295)

[User Interface Specification 6](#_Toc175564296)

[Technical Details 7](#_Toc175564297)

[Developer Work Plan 8](#_Toc175564298)

[Timeline 10](#_Toc175564299)

[Checkpoint #1 (Oct 1st) 10](#_Toc175564300)

[Checkpoint #2 (Oct 22nd) 10](#_Toc175564301)

[Checkpoint #3 (Nov 12th) 11](#_Toc175564302)

[Checkpoint #4 (Dec 3rd): 11](#_Toc175564303)

[Future Enhancements 11](#_Toc175564304)

[Ethical Considerations 12](#_Toc175564305)

[Bibliography 13](#_Toc175564306)

# Project Summary

**QuickSummary** is a browser extension designed to provide users with an efficient way to summarize long web articles and blogs in real-time. Leveraging state-of-the-art natural language processing (NLP) techniques, QuickSummary analyzes and extracts essential information from online content, presenting it in a clear and concise format.

The extension serves students, researchers, professionals, and general readers who frequently encounter lengthy articles online but wish to gain quick insights without reading in full. QuickSummary supports multiple levels of summarization (brief, medium, detailed) and multiple languages, making it a versatile tool for a global audience. By integrating seamlessly into the browser, it offers a user-friendly interface that requires minimal setup and provides instant access to summarized content.

The project utilizes both client-side JavaScript for text extraction and an external NLP API to ensure fast performance and high-quality outputs. By addressing the common problem of information overload, QuickSummary enhances productivity and focus for a wide range of users. The extension is built to prioritize privacy, with all data processing occurring locally on the user's device unless explicitly permitted otherwise.

# Significance

QuickSummary tackles the pervasive problem of information overload in the digital age by leveraging Natural Language Processing (NLP) to provide concise summaries of lengthy web articles and blogs. The project is significant in several areas of Computer Science. First, it applies NLP algorithms to extract essential information from text, a challenging task that involves understanding context, sentiment, and semantics, by using advanced techniques in machine learning and data processing. These technical skills were mainly acquired through internships, online courses, and CSC370: Data Mining.

Second, this project demonstrates expertise in web development by creating a browser extension that seamlessly integrates into users' daily browsing habits. This involves utilizing web technologies like HTML, CSS, React, and JavaScript, along with the Web Extensions API to ensure browser compatibility. The development of an intuitive and responsive user interface emphasizes the importance of user experience (UX) design, making the tool accessible and easy to use.

The project also involves real-time processing and optimization challenges to maintain efficient performance without impacting browser speed, requiring a deep understanding of asynchronous programming, data caching, and API rate-limiting strategies. By combining these elements, QuickSummary exemplifies how Computer Science principles can be applied to solve practical problems.

# Required Tools & Availability

The development of QuickSummary relies on several tools, technologies, and platforms that are readily accessible. The core functionality of the browser extension, including text extraction and user interface development, is implemented using JavaScript, with Visual Studio Code serving as the primary Integrated Development Environment (IDE). HTML and CSS are used to create a visually appealing and user-friendly interface. Additionally, Python is utilized for an optional microservice to handle more advanced NLP processing, leveraging libraries such as spaCy, NLTK, or Transformers from Hugging Face. Access to external NLP services like OpenAI GPT or Hugging Face Transformers API is essential for text summarization, and these APIs are available through online registration and secure API integration.

The development environment is set up with Visual Studio Code version 1.93.1, Node.js, and npm, providing the necessary tools for managing dependencies and building the project. Git and GitHub are used for version control, with repositories hosted on GitHub for collaborative purposes and backup, both of which are already configured on the development machine. For browser extension development, the Chrome WebExtensions API is used, which is compatible with major browsers like Chrome and Firefox, and the Chrome Developer Tools aid in debugging and real-time testing.

Testing the extension in Chrome browsers is performed to ensure the browser compatibility, with the required browser freely available on the development machine. A developer account on the Chrome Web Store is needed to submit the extension for public use, which can be created at a minimal cost. The development is conducted on a personal laptop Dell XPS 13 with OS 11th Edition Version 23H2 with sufficient memory (16 GB RAM) and storage, ensuring that all hardware requirements are met. Stable internet access, necessary for using external NLP APIs and maintaining the development environment, is readily available through home or university networks. Overall, all tools and resources required for this project are either currently available or easily accessible, allowing the development of QuickSummary to proceed smoothly.

# Demonstration Plans

The demonstration of QuickSummary will be conducted live in the Linux Lab in the Julian building during our checkpoint meetings. To ensure everything works as expected, I will practice the demonstration a day prior by visiting the lab and testing the setup. Since the project is web-based and runs entirely on my personal laptop, I will join our Zoom class meeting to share my screen during the presentation.

I will ensure I have all the necessary adapters to connect my laptop to the lab projector if needed.

By preparing these contingency plans, I aim to provide a reliable demonstration of QuickSummary’s features, ensuring that all functionalities are showcased effectively during each checkpoint meeting.

# Qualifications

My professional experience further demonstrates my readiness for this project. During my internship as a Software Engineer at Cummins Inc., I implemented and refined SQL scripts and Scala code to manage and process large datasets efficiently. I also developed a web application using React, Firebase, JavaScript, and the Google Maps API, which required extensive use of JavaScript and frontend development skills—essential for building the QuickSummary browser extension. Additionally, I created a caching system and dynamic DAX measures in Power BI, which honed my skills in optimizing performance and handling large-scale data.

As a Machine Learning and Coding Instructor at iD Tech, I designed and taught Python programming courses focusing on machine learning principles. This experience deepened my understanding of Python and its libraries, such as Pandas, Seaborn, Matplotlib, and scikit-learn, which will be useful if I choose to incorporate a Python-based microservice for advanced NLP processing in QuickSummary.

Furthermore, my role as a Data Analyst at Bridge Builder Strategies involved mastering Python and using data analysis tools like Pandas to cleanse and analyze datasets. I applied advanced statistical methods and synthesized insights into reports with data visualizations, demonstrating my ability to handle data-driven decision-making and presentation—skills that are directly applicable to developing an extension that processes and summarizes web content.

Overall, my academic background, combined with hands-on experience in web development, machine learning, data analysis, and teaching, provides me with the knowledge and skills necessary to successfully complete the QuickSummary project.

# Project Specifications

## Functional Specifications

QuickSummary consists of several key components, each designed to fulfill a specific function:

1. **Text Extraction**: The extension automatically extracts the main content from any web page, excluding unnecessary elements such as advertisements, navigation menus, and user comments. It identifies the primary article body and prepares the text for further processing.
2. **Summarization**: The extension utilizes external NLP APIs (such as OpenAI GPT or Hugging Face Transformers) to generate concise summaries of the extracted text. The user can choose from different summary lengths—short, medium, or detailed—based on their preferences.
3. **Multilingual Support**: QuickSummary offers support for multiple languages, allowing users to summarize content in languages other than English. This feature broadens the extension's usability for a global audience.
4. **Customization Options**: Users can configure settings to adjust the summary length, select their preferred language, and set other personalization preferences. These settings are saved locally, so users do not need to configure them each time they use the extension.
5. **Privacy and Security**: The extension ensures that all data processing occurs locally on the user's device to protect user privacy. Users have the option to share anonymized data to help improve the summarization algorithms, but this is strictly optional.
6. **Error Handling and Notifications**: If the extension encounters unsupported content types or fails to connect to the external NLP APIs, it displays a friendly error message and suggests possible actions, such as refreshing the page or adjusting settings.

## User Interface Specification

The user interface (UI) of QuickSummary is designed to be minimalistic and easy to use, integrating seamlessly into the browser environment:

* **Popup Interface**: The primary interface is a popup window that appears when the extension icon is clicked. This popup includes options for initiating the summarization process, selecting summary length, and choosing the desired language. It also displays the summarized content once generated.
* **Settings Menu**: The popup window includes a settings button that opens a submenu, allowing users to customize their preferences, such as default summary length, language, and other options. The settings menu also provides access to the user feedback feature, where users can rate the quality of summaries and submit feedback.
* **Loading and Error Indicators**: While the extension processes a web page, a loading indicator is displayed in the popup to inform users that the summarization is in progress. If an error occurs, a clear and concise error message will be shown, along with suggestions for resolving the issue.
* **Responsive Design**: The UI is built with responsiveness in mind, ensuring it displays correctly on various screen sizes and resolutions. The design will use CSS for styling and dynamic elements to maintain a consistent look and feel across different browsers.

# Technical Details

QuickSummary will be implemented as a Chrome browser extension, leveraging the WebExtensions API to ensure compatibility across multiple browsers. The development of this extension will involve several key components, each designed to fulfill specific functions necessary for extracting and summarizing web content efficiently.

1. **Text Extraction**: QuickSummary uses JavaScript to extract the main content of a webpage, ignoring non-essential elements like advertisements, navigation menus, and user comments. The content extraction will utilize the Document Object Model (DOM) to identify and isolate the primary article body. This is achieved by targeting common HTML tags and structures, such as <article>, <p>, and <div> with specific classes often used for main content. The tradeoff here is that while this method is fast and efficient, it may not always perfectly isolate content on less conventional or poorly structured web pages. Therefore, the extension includes fallback strategies to handle unexpected page structures, such as searching for the longest text block as a heuristic for main content.
2. **Summarization**: The core summarization functionality is implemented by integrating external NLP APIs, such as OpenAI's GPT or Hugging Face Transformers. The extension sends the extracted text to these APIs via secure HTTP requests, specifying the desired summary length and language. The APIs return a summarized version of the content, which is then displayed in the extension’s popup. The tradeoff in using external APIs is the dependency on third-party services, which may introduce latency or cost concerns. However, these services provide state-of-the-art NLP models that offer superior summarization quality compared to local algorithms, making this choice justified.
3. **Multilingual Support**: QuickSummary supports multiple languages by leveraging the capabilities of the chosen NLP APIs, which offer pre-trained models for various languages. The extension detects the language of the extracted text using a lightweight JavaScript library, franc-min, before sending it to the API with the appropriate language parameter. The tradeoff in using external APIs for multilingual support is the potential cost associated with API usage across large volumes of text. However, this approach simplifies development and leverages existing resources, avoiding the need to build custom language models from scratch.
4. **Customization and User Preferences**: User preferences, such as default summary length and preferred language, are stored locally using the browser’s localStorage API. This method provides a lightweight and straightforward way to persist user settings across sessions without requiring server-side storage or user authentication. The tradeoff here is that data is stored only on the user’s device and cannot be synced across multiple devices or browsers. However, this choice aligns with QuickSummary’s focus on privacy and simplicity.
5. **Privacy and Security**: All data processing, including text extraction and language detection, occurs locally on the user's device to maintain privacy. When summaries are generated, only the minimal amount of necessary text is sent to external APIs, and no personal data is transmitted. Additionally, secure HTTP requests (HTTPS) are used to protect data in transit. A tradeoff of this approach is that some features, like offline summarization, are limited due to the reliance on external APIs. However, this ensures the highest level of privacy for users while still delivering high-quality summaries.
6. **Error Handling and Notifications**: The extension incorporates robust error handling to manage issues like unsupported content types, API failures, or network errors. JavaScript try-catch blocks are used to handle errors gracefully, and clear, user-friendly error messages are displayed in the popup. A notification system is integrated to alert users of errors and suggest possible solutions, such as refreshing the page or modifying settings.
7. **Performance Optimization**: To minimize the impact on browser performance, the extension is designed to use asynchronous JavaScript functions and web workers where necessary, allowing background tasks like text extraction and API requests to run without blocking the main thread. Caching strategies are employed to store recent summaries locally using the browser's IndexedDB to reduce the frequency of API calls, thereby optimizing both speed and cost.

# Developer Work Plan

* **Week of September 17**: Finalize the full project proposal based on feedback. Begin researching existing browser extensions and summarization tools to identify best practices and potential challenges. Set up the development environment, including configuring Visual Studio Code, installing necessary extensions, and setting up the GitHub repository for version control. Start initial planning for the extension’s architecture.
* **Week of September 24**: Begin implementing the basic structure of the extension. Develop the popup interface using HTML, CSS, and JavaScript, and create initial functionality for user interaction. Start coding the text extraction module using JavaScript to identify and extract the main content of a webpage. Conduct initial tests to ensure the extraction module correctly identifies and retrieves the main content from various web pages.
* **Week of September 30**: Finalize the initial version of the text extraction module. Begin integrating an external NLP API (such as OpenAI GPT or Hugging Face Transformers) for summarization. Develop functionality to send extracted text to the API and receive summarized content. Test API integration with sample texts to ensure proper functionality. Prepare for Checkpoint #1 by creating a presentation and backup video of the key features.
* **Tuesday, October 1**: **Checkpoint #1**: Present the basic interface, text extraction module, and initial integration with the NLP API. Gather feedback from the presentation and address any issues raised.
* **Week of October 1**: Incorporate feedback from Checkpoint #1. Continue refining the summarization feature and improving the API integration. Begin implementing customization features, such as allowing users to select summary length (short, medium, or detailed) and language options. Develop a settings menu in the extension popup to enable these customizations.
* **Week of October 8**: Continue working on customization features and improving user interface design. Conduct user testing to gather feedback on the interface and functionality. Begin implementing multilingual support by detecting the language of the extracted text and sending the appropriate language parameter to the NLP API.
* **Week of October 15**: Complete the implementation of multilingual support. Start optimizing the extension’s performance using asynchronous JavaScript functions and caching strategies to minimize the impact on browser speed. Develop error handling and notification features to manage issues like unsupported content types or API failures.
* **Tuesday, October 22**: **Checkpoint #2**: Present the customization options, multilingual support, error handling, and performance optimizations. Gather feedback and address any issues raised.
* **Week of October 22**: Begin working on cross-browser compatibility to ensure that the extension functions properly on Chrome, Firefox, and Edge. Start making any necessary adjustments for compatibility. Continue optimizing the user interface based on feedback from earlier user testing, ensuring it is responsive and accessible.
* **Week of October 29**: Continue refining cross-browser compatibility by testing the extension on different browsers and operating systems to identify and resolve any platform-specific issues. Begin preparing for the third checkpoint by finalizing the cross-browser compatibility and completing additional testing.
* **Week of November 5**: Conduct a comprehensive test plan to verify that all functionality works as expected across all supported browsers. Make final adjustments to the extension’s features as needed. Prepare the presentation materials for Checkpoint #3.
* **Tuesday, November 12**: **Checkpoint #3**: Present cross-browser compatibility and finalized features. Gather feedback and address any remaining issues.
* **Week of November 12**: Start preparing for the final presentation and deployment. Begin packaging the extension according to the Chrome Web Store guidelines and submit it for review. Draft the final project report, incorporating details from the development process and feedback received.
* **Week of November 19**: Address any feedback received from the Chrome Web Store review. Make necessary changes and resubmit if needed. Continue refining the final project report and presentation. Make appointments at the W Center and S Center for additional feedback and support in refining the report and presentation.
* **Week of November 26**: Finalize the project report and presentation materials. Ensure all documentation is complete and up to date. Prepare for the final checkpoint presentation, ensuring all features are ready for demonstration.
* **Tuesday, December 3**: **Checkpoint #4**: Present the final demonstration of QuickSummary, highlighting all completed features and functionalities. Complete any last-minute updates based on feedback from the presentation. Submit the final project report, code, and other deliverables to Moodle.

# Timeline

## Checkpoint #1 (Oct 1st)

* Basic browser extension setup with a functional user interface.
* Development of the text extraction module to identify and extract the main content from a webpage.
* Integration of the initial text extraction functionality into the extension popup.
* **Demonstration**: Navigate to a sample webpage and display the extracted text in the extension popup.

## Checkpoint #2 (Oct 22nd)

* Integration with an external NLP API (e.g., OpenAI GPT or Hugging Face Transformers) to generate summaries.
* Implementation of customizable summary length options (short, medium, detailed) within the user interface.
* Development of the settings menu to allow user customization.
* **Demonstration**: Show generated summaries with different length options for various types of web content.

## Checkpoint #3 (Nov 12th)

* Implementation of multilingual support for summarization, detecting text language and adjusting API parameters accordingly.
* Performance optimizations, including asynchronous JavaScript functions and caching strategies to minimize browser impact.
* Refinement of error handling and notification features for unsupported content types and API failures.
* **Demonstration**: Summarize content in multiple languages, show improved performance, and demonstrate error handling.

## Checkpoint #4 (Dec 3rd):

* Final testing and debugging, including Chrome browser compatibility checks (Chrome).
* Preparation of the extension for submission, adhering to Chrome Web Store guidelines.
* Completion of the final project report and presentation materials.
* **Demonstration**: Test error scenarios (e.g., unsupported page types), demonstrate cross-browser compatibility, and show the final version ready for submission.

# Future Enhancements

* QuickSummary could be extended to support other browsers (Firefox, Safari, etc.).
* Development of a mobile app version for summarizing articles on mobile devices.
* Implementation of offline summarization using pre-trained local models.
* Add a feature for advanced analytics and insights such as time spent on different types of content, topics of interest, or content sources.

# Ethical Considerations

Developing QuickSummary involves several ethical considerations related to both the development process and the use of the product by end-users. Addressing these concerns is crucial to ensure that the extension is designed and implemented responsibly, respecting users' privacy, security, and autonomy.

**Data Privacy and Security**: Since QuickSummary processes text from web pages and interacts with external NLP APIs, it is essential to prioritize data privacy and security. To mitigate privacy concerns, all data processing, including text extraction and language detection, occurs locally on the user's device. Only the minimal amount of necessary text data is sent to external APIs, and no personally identifiable information (PII) is transmitted. Additionally, all communication with external APIs will use secure HTTPS protocols to protect data in transit. QuickSummary will also provide clear information about what data is being sent and why, allowing users to make informed decisions about their data usage.

**Transparency and User Consent**: Transparency is vital in building trust with users. QuickSummary will provide users with clear explanations of how the extension works, what data it collects, and how that data is used. Users will be informed whenever data is sent to external APIs, and they will have the option to opt-out of sharing any data that is not essential for the core functionality of the extension. The extension will also include a detailed privacy policy outlining data handling practices, ensuring users are fully aware of any potential risks.

**Bias in NLP Algorithms**: The use of NLP models for text summarization raises potential ethical concerns regarding bias. AI models trained on large datasets can inadvertently reflect biases present in the training data, potentially leading to biased or misleading summaries. To address this, QuickSummary will select reputable NLP APIs and models known for their efforts to mitigate bias. Additionally, users will be encouraged to provide feedback on the quality and neutrality of the summaries generated, and the extension will incorporate this feedback to improve future results.

# Bibliography

* "Chrome Web Store Developer Documentation." Google Developers, Google, n.d., <https://developer.chrome.com/docs/webstore/>.
* Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805, 2018, <https://arxiv.org/abs/1810.04805>.
* "Ethics Guidelines for Trustworthy AI." High-Level Expert Group on Artificial Intelligence (AI HLEG), European Commission, 2019, <https://ec.europa.eu/digital-strategy/news-redirect/33359>.
* Francois, T., et al. "Development of a Freely Available Wide Coverage Resource for French." Proceedings of the 12th Language Resources and Evaluation Conference (LREC), European Language Resources Association (ELRA), 2020.
* Madsen, Chris. Building Browser Extensions: Hands-On Guide to Developing and Distributing Extensions for Chrome and Firefox, 2nd ed., O’Reilly Media, 2021.
* "OpenAI GPT API Documentation." OpenAI, OpenAI, 2023, <https://platform.openai.com/docs/>
* "WebExtensions API Documentation." Mozilla Developer Network (MDN), Mozilla, n.d., <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions>.