Tartu University

Faculty of Social Studies

Narva College

Study program “Information Technology Systems Development”

Adam Alidibirov

DEVELOPMENT OF AN EXTENSION FOR THE WEB BROWSER GOOGLE CHROME TO PROVIDE REAL-TIME LANGUAGE SUPPORT AND VOCABULARY ENHANCEMENT FOR ESTONIAN LANGUAGE LEARNERS.

Bachelor thesis (8 EAP)

Supervisor: Andre Sääsk, M.Sc.

Narva 2024

Tartu Ülikooli Narva kolledž

Sotsiaalteaduste valdkond

Õppekava “Infotehnoloogiliste süsteemide arendus”

Adam Alidibirov

VEEBISIRVIJA GOOGLE CHROME LISANDMOODULI LOOMINE EESTI KEELE ÕPPIJATELE REAALAJA KEELEABI SAAMISEKS JA SÕNAVARA RIKASTAMISEKS

Bakalaureusetöö (8 EAP)

Juhendaja: Andre Sääsk, M.Sc.

Abstract (English)

This bachelor thesis explores the development of a Google Chrome extension aimed at providing real-time language support and vocabulary enhancement for learners of the Estonian language. The extension addresses the common challenge of looking up unfamiliar words while reading online, which can disrupt the reading flow. By integrating EstNLTK for morphological analysis, Ekilex for dictionary data, and DeepL for translations, the extension allows users to quickly access word definitions, forms, and usage examples without leaving the page. The project highlights the importance of tailored language learning tools and sets a foundation for future research in this area.

Resümee (Estonian)

See bakalaureusetöö uurib Google Chrome'i laienduse arendamist, mis pakub reaalajas keeleabi ja sõnavara rikastamist eesti keele õppijatele. Laiendus lahendab levinud probleemi, kus tundmatute sõnade otsimine veebis lugedes katkestab lugemisvoo. Integreerides EstNLTK morfoloogiliseks analüüsiks, Ekilexi sõnastiku andmete jaoks ja DeepL tõlgeteks, võimaldab laiendus kasutajatel kiiresti juurde pääseda sõnade definitsioonidele, vormidele ja kasutusnäidetele ilma lehelt lahkumata. Projekt rõhutab kohandatud keeleõppevahendite tähtsust ja loob aluse edasisteks uuringuteks selles valdkonnas.

TABLE OF CONTENTS

1. INTRODUCTION

Learning a new language, such as Estonian, can be a challenging task due to its complex grammar system and extensive vocabulary. Estonian learners often struggle with understanding new words when encountered in context, such as while reading news articles or blog posts. Looking up these words in a dictionary can be time-consuming and disruptive to the reading flow, making it harder to fully engage with the language. As an Estonian language learner myself, I experienced firsthand the challenges of constantly having to switch tabs to look up unfamiliar words while reading web articles. This process was disruptive and tiring, often breaking the reading flow and reducing overall engagement with the content. The development of this extension was driven by my personal experience and the desire to create a tool that could address these issues. By allowing learners to look up words and see their forms and meanings without leaving the page, the extension significantly enhances the reading experience and maintains engagement.

This thesis aims to address the problem of the time-consuming and disruptive nature of looking up unfamiliar words in a dictionary while reading Estonian news articles, blog posts, or any text content on the web. The extension will provide quick and easy access to dictionary information and language-specific features, allowing learners to look up unfamiliar words and view their different forms without leaving the page they are reading.

The primary objectives of this thesis are:

1. To investigate existing language learning tools and identify their key features and limitations in supporting Estonian learners.
2. To design and develop a browser extension that incorporates Estonian-specific language resources and natural language processing tools, providing a seamless learning experience.
3. To document the development process, including the challenges encountered and solutions implemented, serving as a resource for future work in this field.

By achieving these objectives, this thesis aims to contribute to the field of Estonian language learning. The proposed browser extension will offer a practical and effective tool for learners,

demonstrating the potential of language-specific technologies in enhancing the language acquisition process. Furthermore, the insights gained from this project will lay the groundwork for future research and development in the area of language learning assistive technologies.

2. RESEARCH

To inform the design and development of the Estonian language learning browser extension, a review of existing language learning tools was conducted. The review focused on browser extensions, as they offer the most relevant functionality for the purposes of this thesis. The primary objective is to analyze only extensions since the thesis centers around providing dictionary support and language learning tools without interrupting the web browsing experience.

2.1. Existing Solutions for Language Learning

2.1.1. Google Dictionary

One of the most widely used extensions is Google Dictionary (Google, n.d.), which allows users to look up definitions of words by double-clicking on them. While useful for general vocabulary acquisition, Google Dictionary has several limitations.

Pros:

- Easy to use, with definitions accessible by simply double-clicking on a word
- Integrates well with web browsing for seamless vocabulary lookup

Cons:

- Primarily supports English, with limited functionality for other languages
- Lacks features such as displaying morphological forms or providing contextual usage examples, which are crucial for learning a morphologically complex language like Estonian
- For Estonian, it only searches for the word in the Google search engine and provides the first found definition, if any, rather than using a dedicated Estonian dictionary

2.1.2. DeepL

Another popular extension is DeepL (DeepL, n.d.), which offers machine translation capabilities. Users can select a piece of text and view its translation in another language.

Pros:

- Provides high-quality machine translations for many language pairs
- Allows users to quickly understand the meaning of foreign language text
- Have dictionaries for popular languages such as English, German, Chinese and etc.

Cons:

- Does not currently support Estonian dictionary lookup
- Focuses on translation rather than providing dictionary definitions, word forms, and usage examples
- Less suitable for the specific needs of Estonian learners looking to deeply understand the language

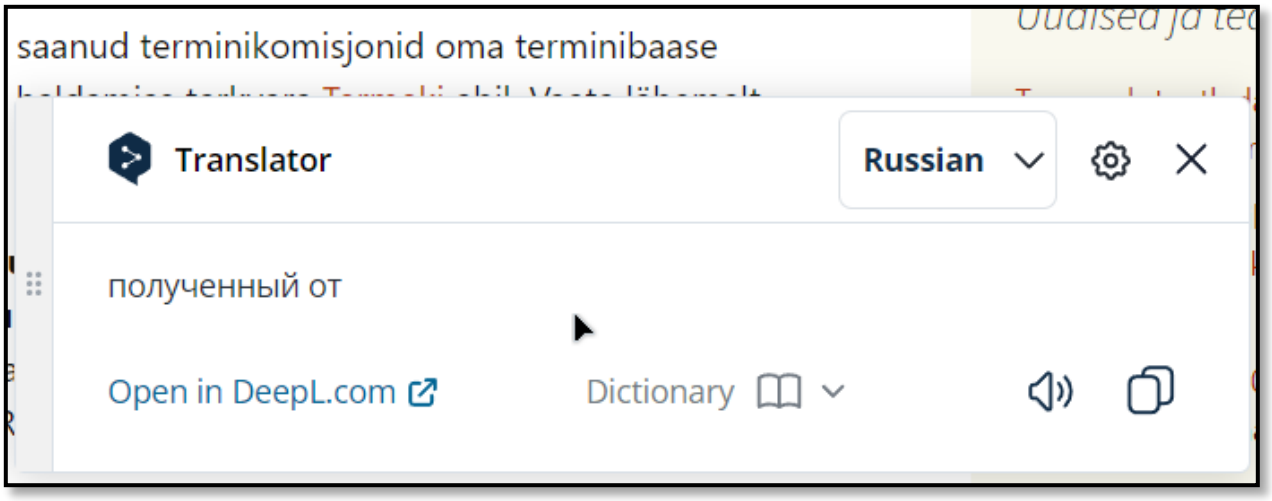

Figure 1 Deepl extension

2.1.3. Definer

Definer (Definer, n.d.) is an extension that offers more comprehensive dictionary features compared to Google Dictionary.

Pros:

- Provides usage examples to illustrate words in context
- Allows users to save words for later review and practice

- Combines various tools like Google Dictionary, translations, DuckDuckGo instant answers, Wikipedia, etc. for an all-in-one language learning experience

Cons:

- Primarily designed for English learners and lacks support for the morphological complexities of Estonian
- Estonian dictionary support is lacking, facing similar issues as Google Dictionary

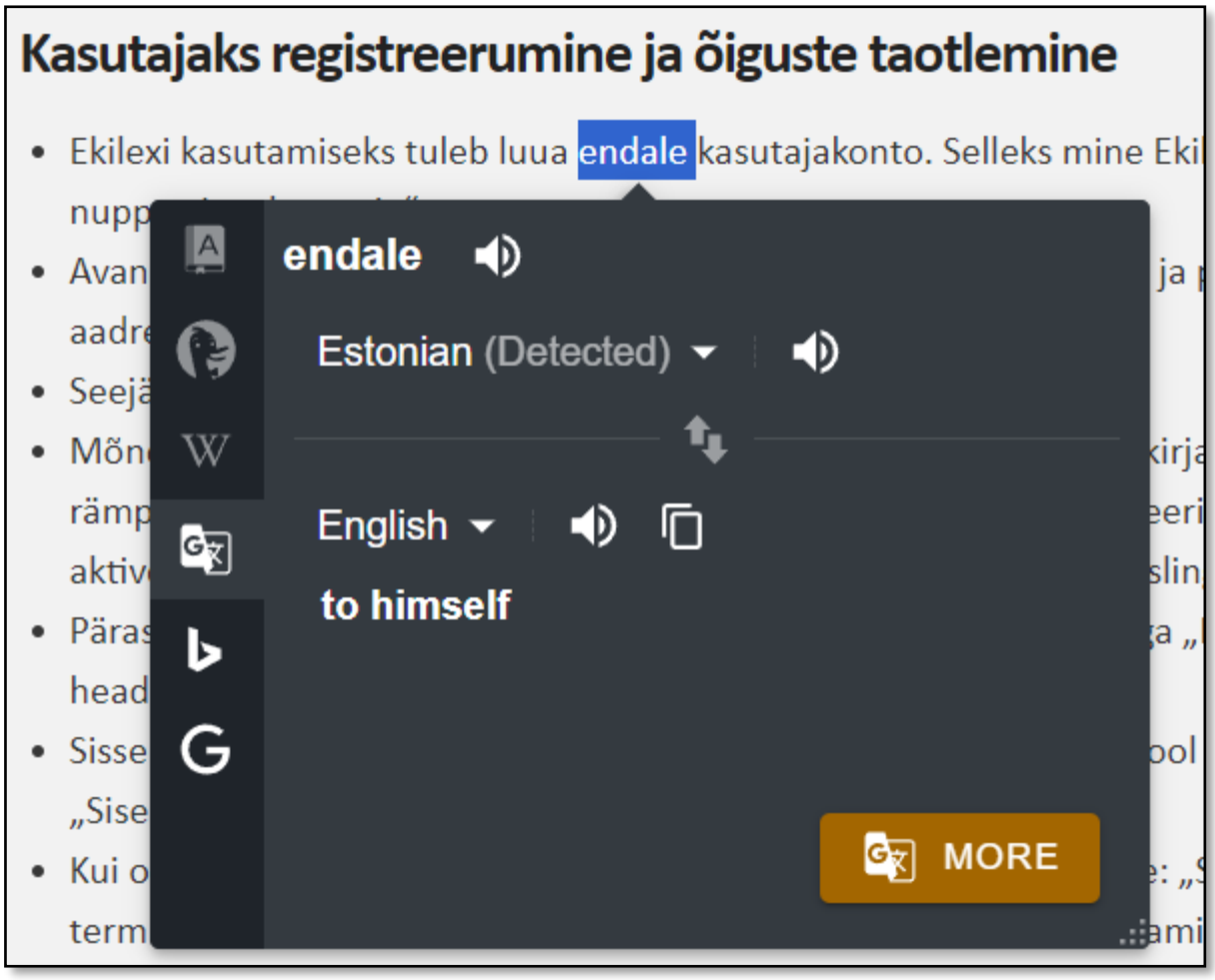

Figure 2 Definer extension

The review of existing language learning browser extensions highlighted a significant gap in the support available for Estonian language learners. While numerous extensions provide

valuable features for general language learning, they primarily cater to English and other widely spoken languages, lacking the necessary functionality for a language like Estonian.

The key findings from the research indicate that current tools:

1. The reviewed extensions allow users to look up definitions or translations without leaving the page they are reading, thereby minimizing interruptions to the reading flow and maintaining user engagement.
2. Existing extensions do not adequately support the complex morphological structures of Estonian, which is crucial for learners to understand word forms and their contextual usage.
3. Most extensions offer general translation and dictionary services but fail to provide detailed language-specific resources tailored to Estonian, such as comprehensive dictionary definitions, usage examples, and morphological analyses, because they don’t use existing Estonian dictionaries and services for language learners.

These findings underscore the necessity for a dedicated Estonian language learning extension. Such an extension should integrate existing natural language processing tools to provide seamless, in-context dictionary lookups, detailed information, and usage examples, thereby addressing the unique challenges faced by Estonian learners. The development of this specialized tool will fill the existing gap and significantly enhance the language acquisition process for Estonian learners.

2.2. Technologies and Tools Used

To develop the Estonian language learning browser extension, different of technologies and tools were used. This section provides an overview of the key components used in the development process.

2.2.1. Backend Development

To develop the backend of the extension was decided to use Python, which is a widely-used programming language. Python was chosen for its extensive ecosystem of libraries and tools, particularly in the domain of natural language processing.

To serve the backend functionality, the FastAPI web framework was used. FastAPI (FastAPI, n.d.) is a modern, high-performance framework for building APIs with Python (FastAPI, 2024). It offers automatic API documentation, type hints for improved development experience, and built-in support for asynchronous programming. The code snippet below demonstrates how concise and elegant it is to create an endpoint for retrieving word information using FastAPI.

```python
@router.post("/info/")
async def get_word_info_view(data: schemas.WordInfoRequest):
    infinitive = await get_infinitive_form(data.text, data.start, data.end)
    words = await eki_search_word(infinitive)
    word_details = await eki_word_details(word)
    return word_details
```

Figure 3 example of FastAPI request handler function (Source: Author)

EstNLTK (EstNLTK, n.d.), an open-source Python library, was utilized for Estonian-specific language processing tasks. This comprehensive library offers a wide array of tools for working with Estonian text, such as tokenization, morphological analysis, and named entity recognition (Siim Orasmaa, 2016). EstNLTK's capabilities enable the extension to provide accurate dictionary forms and grammatical information by detecting the infinitive form and part of speech of a selected word based on its context. This feature is crucial for Estonian, a language with numerous homographs - words that are spelled identically but have different meanings and grammatical roles. For instance, the word "tee" can function as a noun meaning "tea" or "road," or as a verb meaning "to do something." By analyzing the context, EstNLTK determines whether "tee" is being used as a noun or a verb in a given sentence. This allows the extension to make the correct query to the Ekilex API and retrieve the appropriate dictionary entry to display to the user. Thus, EstNLTK's contextual analysis capabilities are vital for providing accurate and relevant information to users of the extension.

The code snippet below illustrates how to obtain the infinitive form of a word using EstNLTK:

```python
async def get_infinitive_form(text, start=None, end=None) -> list[str]:
    morph_tagger = estnltk.taggers.VabamorfTagger(slang_lex=True)

    # Prepare text
    text = estnltk.Text(text)

    # Tag layers required for morphological analysis
    text.tag_layer(['words', 'sentences'])

    # Perform morphological analysis
    morph_tagger.tag(text)

    if start and end:
        word_index = get_index_of_selected_word(text['words'], start, end)
        analysis = text.morph_analysis[word_index]
    else:
        analysis = text.morph_analysis[0]

    return analysis.lemma
```

Figure 4 example of using EstNLTK (Source: Author)

The get_infinitive_form function is an asynchronous function that takes three parameters:

- text: The input text containing the word for which the infinitive form needs to be determined.
- start (optional): The starting index of the selected word within the input text.
- end (optional): The ending index of the selected word within the input text.

The function returns a list of strings representing the infinitive form(s) of the selected word or the first word in the input text if no specific word is selected.

Step by step example:

1. The function initializes an instance of the VabamorfTagger from the EstNLTK library, which is used for morphological analysis of Estonian text. The slang_lex=True parameter enables the tagger to handle slang and colloquial language.

2. The input text is wrapped in an EstNLTK Text object, which provides a convenient interface for text processing and annotation.

3. The function tags the required layers for morphological analysis using text.tag_layer(['words', 'sentences']). This step identifies the word and sentence boundaries in the input text.

4. The morph_tagger.tag(text) line performs the actual morphological analysis on the input text. This process annotates each word with its lemma (base form), part of speech, and other morphological properties.

5. If both start and end indices are provided, the function assumes that a specific word is selected within the input text. It calls the get_index_of_selected_word function (explained in 3.3 section) to determine the index of the selected word within the list of words identified by the tagger. The morphological analysis for the selected word is then retrieved using text.morph_analysis[word_index].

6. If start and end indices are not provided, the function assumes that no specific word is selected(it happens when user selection include only one word without context). In this case, it simply retrieves the morphological analysis for the first word in the input text using text.morph_analysis[0].

7. Finally, the function returns the lemma (infinitive form) of the selected word or the first word, depending on the scenario.

To access comprehensive dictionary data, the extension integrates with the Ekilex API. Ekilex is a database of Estonian dictionaries maintained by the Institute of the Estonian Language (Ekilex ja kirjete koostamine, 2024). The API is built on a REST (Representational State Transfer) architecture, which provides a simple and standardized way for applications to communicate with the Ekilex database over HTTP (Chapter 5: Representational State Transfer (REST), 2021). While the Ekilex API offers the ability for authorized users to become maintainers and edit the dictionary content, this functionality is not utilized by the extension, as it only needs to retrieve information from the database. The API returns data in a structured JSON (JavaScript Object Notation) format, which is easy for the extension to parse and extract the relevant information. Through the Ekilex API, the extension gains access to a wealth of

linguistic data, including definitions, usage examples, and morphological information for Estonian words. This integration allows the extension to provide users with accurate and up-to-date dictionary content, enhancing the language learning experience.

To support on-demand translation of Estonian text, the extension integrates the DeepL API (DeepL, n.d.), a leading provider of high-quality machine translation services for a variety of language pairs. Although translation is not a core feature of the extension, the ability to quickly translate words or phrases can be a valuable supplement to the dictionary functionality. Among the various solutions available for translating text, DeepL was chosen for its free tier (with some limitations), ease of integration, and the availability of Python packages that facilitate its implementation.

The code snippet below demonstrates how to use the DeepL API for translation:

```
translator = deepl.Translator(auth_key='SECRET')

result = translator.translate_text(
    'Kui teed teed, tee mulle ka.',
    target_lang='en',
    source_lang='et'
)

result.text # Output: If you're making tea, make some for me.
```

Figure 5 example of using Deepl package (Source: Author)

In this example, the DeepL translator is initialized with an authentication key. The translate_text method is then called with the Estonian text “Kui teed teed, tee mulle ka.”, specifying the target language as English (‘en’) and the source language as Estonian (‘et’). The API returns the translated text, which is accessed through the text attribute of the result object.

2.2.2. Frontend Development

The frontend of the Estonian language learning browser extension plays a crucial role in providing a seamless and interactive user experience for language learners. To achieve this,

careful consideration must be given to the technologies and architectures used in the frontend development process.

After evaluating various frontend frameworks and libraries, Vue.js emerges as a suitable choice for building the extension's user interface. Vue.js is a progressive JavaScript framework that promotes component-based development, making it easier to create reusable and maintainable code. Its reactive data model and efficient rendering mechanism enable the creation of dynamic and interactive user interfaces (The Progressive JavaScript Framework, 2024).

To streamline the styling process and ensure a consistent visual design, the Tailwind CSS framework will be employed. Tailwind CSS provides a set of utility classes that can be easily applied to HTML elements, allowing for rapid development of custom styles without writing extensive custom CSS. (Hernandez, n.d.) This approach promotes a modular and maintainable CSS codebase, making it easier to style the extension's components.

One of the key requirements for the extension's frontend is the ability to inject its functionality into any web page visited by the user. This allows the extension to capture text selection events and provide word information and language learning features seamlessly. To achieve this, the extension will leverage the Google Chrome extension manifest version 2.

The extension manifest file (manifest.json) will specify a content script that will be injected into every web page visited by the user. The content script will establish communication between the web page and the extension, enabling the capture of text selection events and the injection of the extension's user interface.

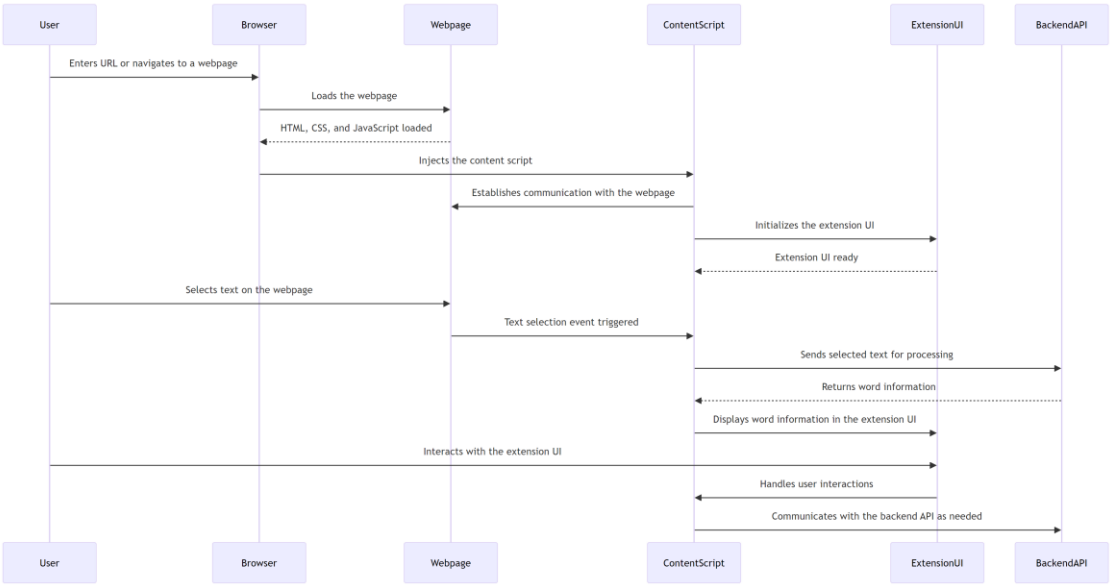

Figure 6 sequence diagram of extension related processes (Source: Author)

By following this architecture and leveraging the power of Vue.js, Tailwind CSS, and the Google Chrome extension manifest, the frontend of the Estonian language learning browser extension will provide a seamless and interactive experience for language learners. The extension will be able to capture text selection events, retrieve relevant word information from the backend API, and display it to the user in a user-friendly and visually appealing manner.

3. PRACTICAL DEVELOPMENT

3.1. Extension Architecture

The Estonian language learning browser extension consists of several interconnected components that work together to provide a seamless user experience. This section provides an overview of the extension's architecture and the role of each component.

At the core of the extension is the backend server, built with Python and FastAPI. The backend server is responsible for handling requests from the extension frontend, processing Estonian text using EstNLTK, and retrieving dictionary and translation data from the Ekilex and DeepL APIs.

The extension frontend is implemented as a Chrome browser extension, using Vue.js for the user interface and Tailwind CSS for styling. The frontend communicates with the backend server via HTTP requests, sending user input (such as selected text) and receiving processed data (such as dictionary definitions and morphological forms).

To interact with web pages, the extension uses Chrome's content script mechanism. A content script is a JavaScript file that runs in the context of a web page, allowing the extension to access and manipulate the page's content. In the case of the Estonian language learning extension, the content script is responsible for detecting when the user selects text on a page and sending that text to the backend server for processing.

The content script also handles the display of the extension's user interface. When the user selects text, the content script creates a floating panel near the selection, which contains the dictionary information and language learning features provided by the extension. This panel is implemented using the Shadow DOM API, which allows the extension to encapsulate its styles and avoid conflicts with the styles of the host web page.

The overall flow of data and interactions between the various components(layers) of the extension is illustrated in the sequence diagram shown in Figure 5. The diagram depicts the following steps:

1. The user selects a word in the browser.

2. The browser extension sends the selected word and its surrounding sentence (if available) to the backend server.
3. The backend server processes the text using EstNLTK to obtain the infinitive form of the selected word.
4. The backend server checks if the word info related to the infinitive exists in the cache (database).
    - If the word information is found in the cache, the cached word information is returned to the backend server.
    - If the word information is not in the cache, the backend server requests the word ID from the Ekilex API using the infinitive. If the word ID is found, the backend server then requests the word definition and usage examples from the Ekilex API using the word ID. The received word information is saved to the cache.
5. The backend server returns the processed text, definition, and usage examples to the browser extension.
6. The browser extension displays the definition in a pop-up panel to the user.
7. The browser extension requests translations for the usage examples from the backend server.
8. The backend server sends the usage examples to the DeepL API for translation.
9. The DeepL API returns the translated usage examples to the backend server.
10. The backend server sends the translated usage examples to the browser extension.
11. The browser extension displays the translated usage examples in the pop-up panel to the user.

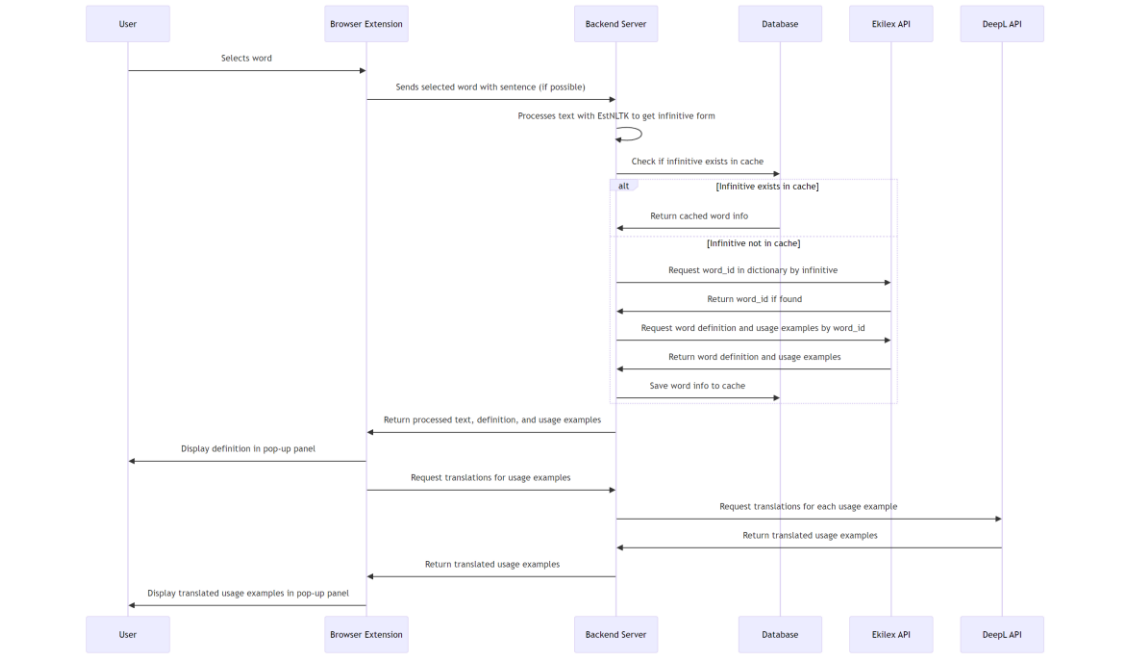

Figure 7 sequence diagram of the Estonian language learning extension (Source: Author)

Overall, the architecture of the Estonian language learning browser extension is designed to be modular, extensible, and efficient. By separating concerns between the backend server, frontend extension, the extension can be easily maintained and updated over time. The use of modern web technologies and best practices ensures a high-quality user experience and good performance, even on resource-constrained devices.

3.2. User Interface Design

The user interface of the Estonian language learning browser extension was designed with a focus on simplicity, clarity, and ease of use. The primary goal was to provide learners with quick access to dictionary information and language learning features without overwhelming them with extraneous details or complex interactions.

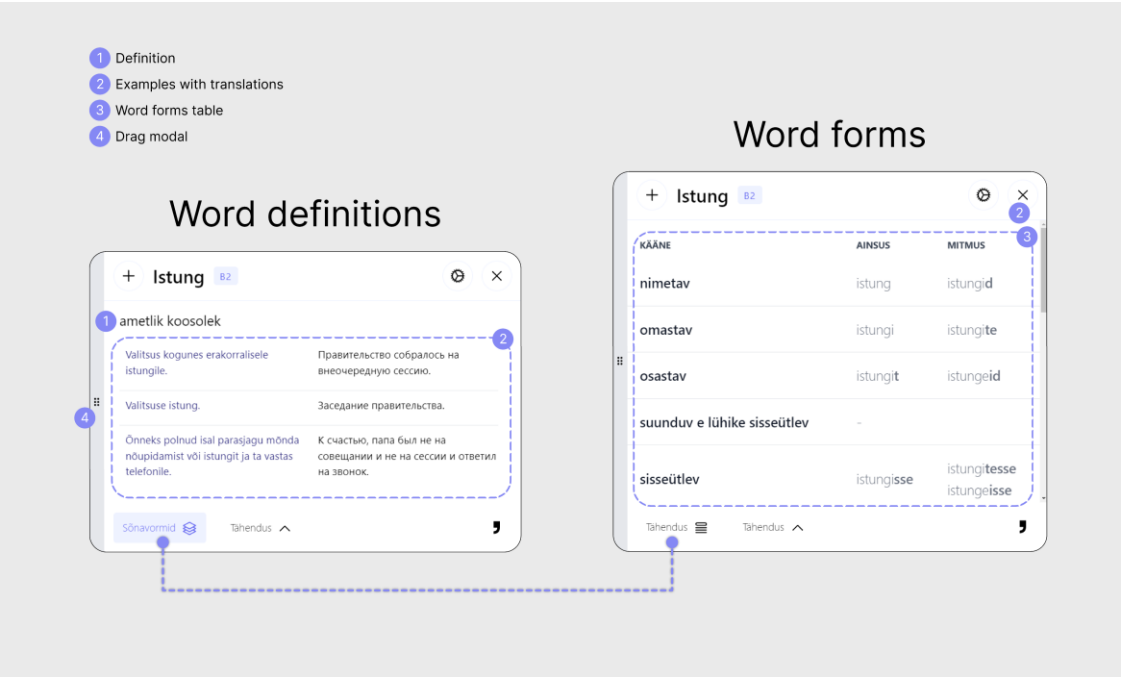

Figure 8 design of the extension modal (Source: Author)

The main component of the user interface is the floating panel that appears when the user selects text on a web page. This panel is divided into several sections, each presenting a different type of information or functionality.

At the top of the panel is the selected word itself, displayed in a large, readable font. Below the word is its primary definition, sourced from the Ekilex dictionary database. The definition is presented with any additional information (such as part of speech) displayed alongside the definition text.

Below the definition section is a table displaying usage examples, also retrieved from Ekilex. The table's first column presents an example of the word's usage in Estonian, while the second column offers a translation of the example into the user's chosen language.

In the bottom of the component located buttons. One of them is a button that shows the morphological information section. This section presents the various grammatical forms of the selected word, organized in a table format. For nouns and adjectives, the table displays singular and plural forms for each of the Estonian grammatical cases. For verbs, the table

shows the conjugated forms for different tenses and moods. The morphological information is presented in a compact, easy-to-scan format, allowing learners to quickly grasp the patterns and variations of the word. In addition to the dictionary and morphology sections, the user interface also includes several interactive features. These include buttons for saving the selected word to a personal vocabulary list, requesting a translation of the usage example, and navigating to more detailed dictionary.

The overall visual design of the user interface is kept simple and clean, with a focus on readability and usability. The color scheme is muted and neutral, ensuring that the extension does not distract from the content of the host web page. Typography is chosen for clarity and legibility, with appropriate contrast and sizing for comfortable reading. Interaction design principles are also carefully considered, with a focus on minimizing friction and cognitive load for the user. Buttons and controls are clearly labeled and intuitively positioned, with appropriate hover and focus states to indicate interactivity. Animation and transition effects are used sparingly, primarily to provide feedback and guide attention.

Throughout the design process, user feedback and usability testing were conducted to iteratively refine and improve the user interface. The resulting design is a balance of functionality and simplicity, providing Estonian language learners with a powerful yet intuitive tool for enhancing their language acquisition journey.

3.3. Challenges and Solutions

The development of the Estonian language learning browser extension presented several challenges, both technical and linguistic in nature. This section outlines some of the key challenges encountered and the solutions devised to address them.

One challenge was the integration of the various APIs and libraries used by the extension. The Ekilex API, used for retrieving dictionary data, and the DeepL API, used for translations, each have their own authentication requirements, data formats, and usage constraints. Integrating these APIs into the extension’s backend server required careful study of their documentation and the implementation of robust error handling and retry logic to ensure reliable operation.

Another challenge was the accurate parsing and analysis of Estonian text using the EstNLTK library. While EstNLTK provides a powerful set of tools for working with Estonian, its usage is not always straightforward, particularly for developers not deeply familiar with linguistic concepts. EstNLTK's analysis returns a list of word objects, and to find the specific word selected by the user, a custom function was developed:

```
def get_index_of_selected_word(words, selection_start: int, selection_end: int) -
> int | None:
    """
    Determines the index of the word in 'words' that overlaps with the given
selection range.
    """
    for idx, word in enumerate(words):
        if selection_start < word.end and selection_end > word.start:
            return idx

    return None
```

Figure 9 example of code to get index of selected word (Source: Author)

This function takes the user's selected word (defined by start and end integers) and the list of words returned by EstNLTK, and returns the index of the selected word in EstNLTK's output.

On the frontend side, a major challenge was the development of the floating panel user interface and its integration with web pages. The use of the Shadow DOM API to encapsulate the extension's styles and avoid conflicts with host page styles proved to be complex, particularly when dealing with edge cases and cross-browser compatibility issues (Using shadow DOM, 2024). While the Shadow DOM API can encapsulate the extension styles, website styles and overridden browser default styles for elements like 'div', 'p', 'h1', 'h2', etc. can still affect the extension. This problem was not completely fixed. Here's an example of how mounting the Vue app looks without and with the Shadow DOM API:

Without Shadow DOM API:

```
const app = new Vue({
  render: h => h(App)
}).mount('#extension-app');
```

Figure 10 vue.js mounting (Source: Author)

With Shadow DOM API:

```
const shadowRoot = document.createElement('div').attachShadow({ mode: 'open' });
const app = new Vue({
  render: h => h(App)
}).mount(shadowRoot);
document.body.appendChild(shadowRoot.host);
```

Figure 11 vue.js mounting with shadow dom api (Source: Author)

From a design perspective, the main challenge was fitting a large amount of information into the small extension modal while maintaining clarity and usability. This required careful selection of the most essential data to display and the omission of less critical details. As an Estonian language learner myself, I focused on the types of information that I found most helpful, such as word forms and usage examples. For instance, a maximum of three usage examples are shown for each word meaning to keep the interface concise and manageable.

In developing this extension, I incorporated design decisions from the researched extensions. For instance, to indicate that the extension dialog is movable, I added a panel displayed on the left side of the extension dialog, similar to the DeepL extension (Figure 1). The button locations were also inspired by DeepL, as they appear user-friendly and easy to read. However, for displaying word forms and meanings correctly and in an easy-to-read manner, I followed the layout and design principles of Sõnaveeb (BNS, 2019).

Figure 8 illustrates the final design of my extension, showcasing how these principles were implemented to create a user-friendly and effective tool for Estonian language learners. The interface is designed to provide quick access to essential information without overwhelming the user, maintaining a balance between comprehensive language support and a clean, navigable interface.On the user experience side, close collaboration with language learners

was maintained throughout the development process. Regular reviews and feedback were conducted to validate the usefulness and usability of the extension. One key feature implemented based on user feedback was the ‘bubble’ button, which shows a small, unobtrusive button when text is selected, rather than immediately displaying the full dictionary modal. This allows users to select text for other purposes without being disturbed by the extension and just touch the bubble when they want to get information about selected word.

Another linguistic challenge encountered was the handling of homographs - words that are spelled the same but have different meanings and grammatical functions. For example, in Estonian, “tee” can mean “tea” (noun), “road” (noun), or “to do something” (verb). The extension’s ability to understand the context and determine the word’s lemma is a powerful feature. However, when the lemma is obtained, words with the same spelling but different meanings (e.g., “tee” as “tea”, “tee” as “road”) pose a challenge when querying the Ekilex API for word information.

Currently, the extension retrieves a list of words based on the lemma, but it always selects the first one, disregarding the potential for multiple meanings. In contrast, Sõnaveeb allows users to choose which meaning to display. Two potential solutions were identified to address this issue:

1. Implement a user interface similar to Sõnaveeb, where the user is presented with the different meanings of the word and can select the appropriate one. While this solution would provide users with the ability to disambiguate the word’s meaning, it may not be the most user-friendly approach.
2. Develop an AI model that can determine the correct meaning based on the context in which the word appears. Thanks to the latest advancements in AI technology, particularly in natural language processing, creating such a model is now a realistic possibility. By leveraging state-of-the-art AI techniques, the extension could automatically select the appropriate meaning without requiring user intervention, greatly enhancing the user experience.

Implementing an AI-based solution would require additional time and resources compared to the first option. However, the benefits of providing accurate, context-specific information to

language learners would be substantial. As AI technology continues to evolve, developing such a model becomes increasingly feasible, making it a promising avenue for future improvements to the extension.

Resolving this challenge through advanced AI techniques would represent a significant step forward in the extension’s ability to provide a seamless and effective learning experience for users.

4. CONCLUSION

In conclusion, this thesis has successfully achieved its primary objectives by researching existing extensions for language learners and developing a new browser extension that supports Estonian language learners. The extension leverages state-of-the-art natural language processing tools and integrates with comprehensive language resources to provide learners with quick and easy access to dictionary information and language-specific features.

Throughout the development process, several challenges were encountered, such as integrating various APIs, accurately parsing and analyzing Estonian text, and designing a user-friendly interface. However, through research, collaboration with language learners, and iterative testing, solutions were implemented to overcome these challenges.

One significant observation during the investigation of existing language learning tools was that most tools are designed primarily for English learners. Estonian learners, however, require more detailed information about word forms and usage examples due to the complexity of Estonian grammar. This need highlights the importance of providing relevant information, such as different word forms and their uses, directly within the reading environment.

The resulting extension offers seamless learning experience, allowing users to look up unfamiliar words and view their different forms without disrupting their reading flow. The extension's ability to understand the context and determine the appropriate lemma for a word is a powerful feature that sets it apart from existing language learning tools. However, while the extension can distinguish between different parts of speech for the same word, it currently cannot disambiguate between homographs (words that are spelled the same but have different meanings and the same part of speech). Additionally, this is the unique language learning extension designed specifically for the Estonian language.

Future development could focus on implementing an AI model to accurately determine the correct meaning of a word based on its context. This would greatly enhance the user experience by eliminating the need for manual disambiguation of homographs. By leveraging state-of-the-art AI techniques, the extension could automatically select the appropriate meaning of a word, providing learners with more accurate and context-specific information.

This advancement would allow learners to see contextually relevant meanings rather than just a list of possible definitions, making the learning process more intuitive and effective.

In summary, this thesis has made a contribution to the field of Estonian language learning by developing a practical tool that demonstrates the potential of language-specific technologies in enhancing the language acquisition process. The insights gained from this project will serve as a foundation for future research and development in the area of language learning assistive technologies, paving the way for even more advanced and personalized learning experiences.

5. REFERENCES

BNS. (2019, 02 11). Eesti Keele Instituudis valmis uus sõnastikuportaal Sõnaveeb. Retrieved from Postimees: https://www.postimees.ee/6520282/eesti-keele-instituudis-valmis-uus-sonastikuportaal-sonaveeb

Chapter 5: Representational State Transfer (REST). (2021, 05 13). Retrieved from UC Irvine Donald Bren School of Information & Computer Sciences: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Ekilex ja kirjete koostamine. (2024). Получено из Eesti terminitöö: https://terminoloogia.ee/ekilex-kirjete-koostamine/

ERR. (2024, 2). Ekilex. Retrieved from Ekilex: https://keeleinstituut.github.io/ekilex/

FastAPI. (2024). Retrieved from FastAPI: https://fastapi.tiangolo.com/

Hernandez, I. (n.d.). Your Complete Tailwind CSS Primer. Retrieved from DreamHost: https://www.dreamhost.com/blog/tailwind-css/

Siim Orasmaa, T. P.-J. (2016, 08 30). EstNLTK - NLP Toolkit for Estonian. Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 2460–2466). European Language Resources Association (ELRA). Retrieved from GitHub: https://github.com/estnltk/estnltk

The Progressive JavaScript Framework. (2024). Retrieved from Vue.js: https://vuejs.org/about/faq.html

Using shadow DOM. (2024). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_shadow_DOM

6. APPENDIX

6.1. Licence

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, ____Adam Alidibirov_____,
(autori nimi)

1. Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
Veebisirvija Google Chrome lisandmooduli loomine eesti keele õppijatele reaalaja keeleabi saamiseks ja sõnavara rikastamiseks_____,
(lõputöö pealkiri)

mille juhendaja on ____Andre Sääsk_____,
(juhendaja nimi)

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Adam Alidibirov
19.05.2024