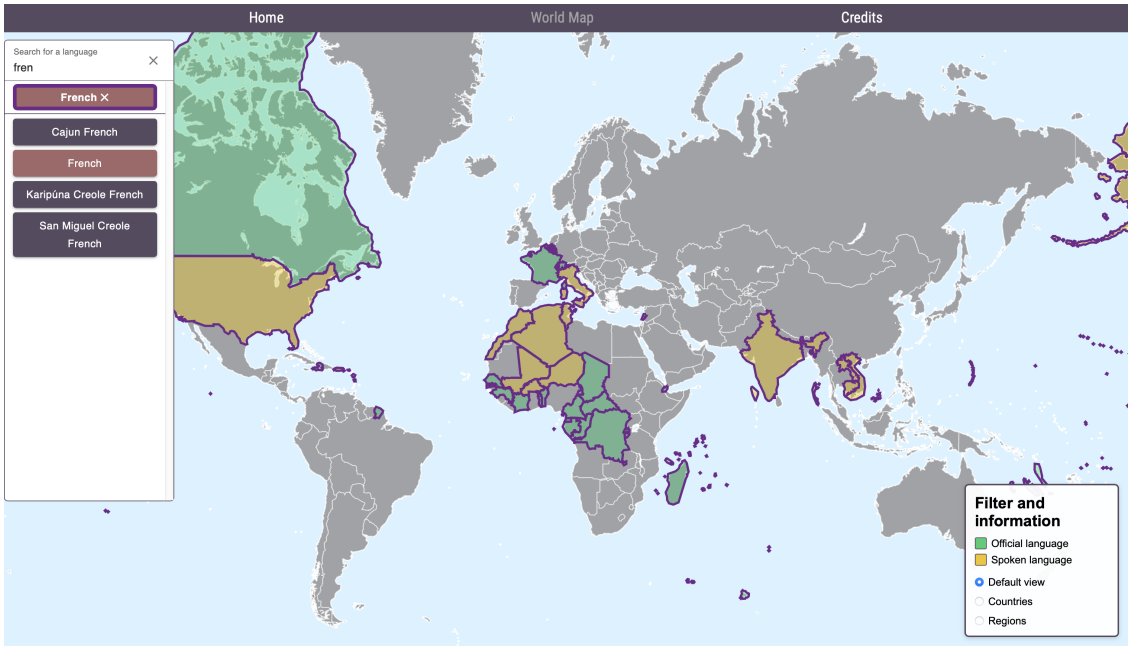
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



An Interactive Map of the World's Languages

Bachelor's thesis in Computer science and engineering

EMMA AXELSSON
BENJAMIN ELD
ERIK GREEN BLOMROOS
FELIX GUSTAVSSON JONSSON
HANNA MAGNUSSON
PONTUS WIKSTRÖM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Bachelor's thesis 2025

An Interactive Map of the World's Languages

EMMA AXELSSON
BENJAMIN ELD
ERIK GREEN BLOMROOS
FELIX GUSTAVSSON JONSSON
HANNA MAGNUSSON
PONTUS WIKSTRÖM

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

An Interactive Map of the World's Languages

EMMA AXELSSON BENJAMIN ELD ERIK GREEN BLOMROOS
FELIX GUSTAVSSON JONSSON HANNA MAGNUSSON PONTUS WIKSTRÖM

Supervisor: Krasimir Angelov, Department of Computer Science and Engineering
Examiner: Patrik Jansson, Department of Computer Science and Engineering
Graded by teacher: Yehia Adb Alrahman, Department of Computer Science and Engineering

Bachelor's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The front cover shows the interactive language map with French selected. Areas where French is spoken are displayed on the map.

Typeset in LaTeX
Gothenburg, Sweden 2025

An Interactive Map of the World's Languages
EMMA AXELSSON, BENJAMIN ELD, ERIK GREEN BLOMROOS,
FELIX GUSTAVSSON JONSSON, HANNA MAGNUSSON, PONTUS WIKSTRÖM

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This project thesis presents the development of an interactive language map that visualizes the global distribution of spoken languages. The system integrates linguistic data from Wikidata and geospatial data from OpenStreetMap to allow users to search for languages and view where they are spoken. The application was built with a React and TypeScript frontend, a NestJS backend, and a MongoDB database hosted in the MongoDB Atlas cloud. SPARQL queries were used to extract structured language data, which was cleaned and stored to enable fast queries. The resulting product enables interactive visualization of language regions with support for filtering by country or region, viewing speaker statistics, and language family information. The map also supports interactive exploration by clicking on a country to display information about the languages spoken in that country. The project demonstrates how open data sources can be used to create educational visualizations, although the quality and coverge of the external databases constrain the final accuracy.

Sammandrag

Denna projektrapport presenterar utvecklingen av en interaktiv språkkarta som visualiserar den globala fördelningen av talade språk. Genom att integrera språklig information från Wikidata med geospatial data från OpenStreetMap möjliggörs sökning efter språk och visualisering av var de talas i världen. Applikationen är byggd med en frontend i React och TypeScript, en backend i NestJS samt en molnbaserad MongoDB-databas via MongoDB Atlas. Språkinformationen hämtades med hjälp av SPARQL-frågor och bearbetades sedan innan den lagrades för att möjliggöra snabb åtkomst. Den färdiga produkten erbjuder interaktiv visualisering av språkområden med stöd för filtrering efter land eller region, visning av antal talare samt information om språkfamiljer. Kartans interaktiva funktionalitet gör det möjligt att klicka på ett land för att visa detaljerad information om vilka språk som talas där. Projektet visar hur öppna datakällor kan användas för att skapa informativa och lärorika visualiseringar, även om datakvaliteten och täckningen i externa källorna begränsar noggrannheten.

Keywords: Interactive Map, Language Visualization, Wikidata, OpenStreetMap, SPARQL, GeoJSON, React, NestJS, MongoDB Atlas, Linguistic data

Glossary

API Application Programming Interface, a set of rules that allows programs to communicate with each other. 7, 9

BSON Binary Javascript Object Notation, a binary-encoded serialization of JSON documents. 21

CSS Cascading Style Sheets, a language used to adjust the style and layout of HTML elements. 15

GIS Geographic Information System, a general term describing systems or frameworks used for gathering, managing, and analyzing spatial and geographic data. 6, 40

HTTP Hypertext Transfer Protocol, a protocol used to transfer data over the web enabling communication between clients and servers. 7, 22

JSON JavaScript Object Notation, a lightweight data-interchange format that is easy for both humans and computers to read and write. 7–9, 21

KML Keyhole Markup Language, an XML-based format for representing geographic data for applications such as Google Earth. 40

MUI Material User Interface. 15

OSM OpenStreetMap. 2, 6–9, 12, 21

RGB Red, green and blue. 15

SPARQL SPARQL Protocol and RDF Query Language, used to query and manipulate data stored in an RDF format. 7–9

UI User Interface, the part of a system that a user interacts with directly. 14, 15, 17

URL Uniform Resource Locators, a web address that provides a unique, specific location for a particular resource on the internet. 7

XML Extensible markup language. 40

Contents

List of Figures

1

Introduction

A language map is a thematic map that depicts the geographic distribution of language speakers, a vital tool used in the field of linguistics. The practice of visually categorizing the world's languages is often attributed to the German linguist Gottfried Hensel, whose maps date back to the 18th century [1]. Unlike political maps depicting clearly defined national borders, language does not adhere to fixed geographical boundaries. Instead, languages are tied to ethnic groups and cultural communities that often span multiple disconnected regions. For example, English is spoken in several countries, in some as a mother tongue and others as a second language, making it difficult to represent on a map with simple territorial boundaries. Multilingual regions further complicate the picture, as multiple languages are spoken in overlapping regions. A static paper map struggles to capture the true shape of real-world linguistic landscapes because of these challenges [2].

Some of the first attempts to digitally map language data began in the 1970s. Notable examples include the Computer Developed Linguistic Atlas of England, the Atlas Linguarum Europae, and the Kleiner Deutscher Sprachatlas, which are considered the first digital linguistic atlases [3]. Since then, modern projects have further refined the field, yet finding a free, comprehensive, and well-visualized digital language map remains surprisingly hard. For example, Ethnologue, which brands itself as the "...world's most comprehensive catalog of languages", restricts full access behind a paywall [4], [5]. Meanwhile, Glottolog, a free alternative specializing in language family trees, offers rich language data, but remains poorly visualized map-wise [6]. The site Muturzikin provides a relatively comprehensive language map, but is dated, poorly sourced, and closer to a static map than an interactive tool [7]. Finally, Wikipedia contributors have created a static map of the world language distribution, but it is low resolution and contains insufficient citations of sources [8].

Language maps are useful for curious individuals with an interest in linguistics, as well as for scholars in linguistics and anthropology who seek an easy and user-friendly tool for analyzing the global linguistic distribution. Open-source and user-friendly visualization tools can also assist researchers by providing structured and easily digestible linguistic data. In addition, institutions that promote linguistic diversity and cultural heritage, such as educational organizations and language preservation initiatives, can potentially benefit from a visually engaging tool that highlights lesser-known languages and cultures.

1.1 Purpose

The purpose of this project is to develop an interactive language map that visualizes the geographic distribution of languages spoken worldwide. By integrating linguistic and geospatial data, it addresses the lack of high-quality, interactive language maps.

1.2 Objective

The objective of the project is to provide users with a tool that enables them to:

- Search for and explore individual languages.
- View the geographic distribution of languages based on open geospatial data.
- Access key metadata about languages, such as the number of speakers and language family.

1.3 Scope

This project aims to develop a functional prototype for visualizing language data, but certain limitations apply due to the available resources. The following subsections outline the key limitations.

1.3.1 Sourcing Data

The primary data source for the project is Wikidata [9]. Wikidata is openly available and frequently updated, making it suitable for this project. As a crowd-sourced platform, Wikidata carries the risk of inconsistencies or gaps in coverage. However, unlike official census data, it is not subject to political censorship or selective omission of minority languages, as seen in certain authoritarian regimes [10]. While combining multiple datasets could improve data completeness and accuracy, it could also introduce significant complexity in terms of data integration, schema alignment and potential inconsistencies between sources. To maintain coherence and reproducibility, the project limits itself to a single linguistic data source.

Creating a world language map requires maps of the world's various regions. For this project, the map database OpenStreetMap (OSM) is used as it provides access to polygon data for countries and regions, which are necessary to create the visual map [11]. Many Wikidata entries include OSM identifiers, making the integration of the two datasets straightforward without requiring additional geographic sources.

1.3.2 Dialects

Dialects would be valuable in the interactive map, but are excluded to keep the project manageable. Wikidata does not contain data surrounding dialects with the same completeness as it does with languages. Including dialects would therefore require integrating an additional database, which would conflict with the project's defined scope of using only one linguistic data source. Furthermore, distinguishing between a language and a dialect is often subjective and varies between sources [12]. Including dialects could therefore increase the ambiguity and reduce the accuracy of the map.

2

Methods

This chapter outlines the methods and technologies used in the development of the interactive language map. It is divided into sections covering the database, backend architecture, frontend development, and user testing. Each section specifies the tools, frameworks, and bibliographies that form the foundation of the project.

Figure 2.1 provides an overview of the system's architecture and illustrates how the frontend, backend, external databases, and project database interact to support the functionality of the interactive language map.

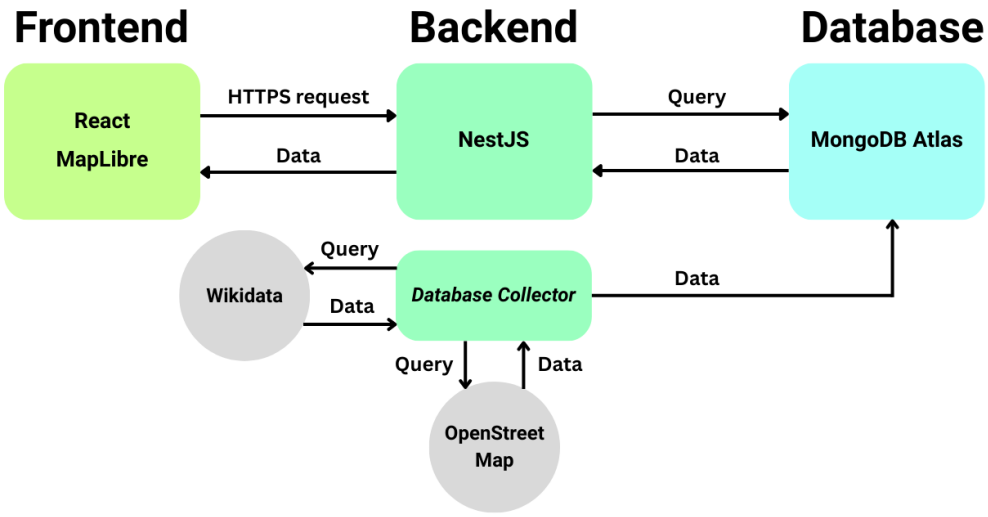


Figure 2.1: System architecture showing the interaction between frontend, backend, external databases, and the project database.

2.1 Database

This section outlines how language and geographical data were sourced, processed, and stored for use in the interactive language map. It describes the choice of external data sources, the method used to fetch and clean the data, and the decision to store the final dataset in a cloud-based database.

2.1.1 Wikidata and Sourcing Metadata

The data source for language metadata is Wikidata, a free and collaboratively maintained knowledge base operated by the Wikimedia Foundation, a nonprofit organization [9]. Designed as a central repository for structured and machine-readable data of the world's knowledge, it provides several projects including Wikipedia with their data [9]. Wikidata is built upon crowd-sourced data, meaning anyone can add or edit data anytime. The collaborative approach to the project has made it one of the most comprehensive knowledge resources currently available today [13].

The quality of knowledge stored in Wikidata is comparatively more reliable in terms of both quality and quantity compared to other online databases, according to Faeber et al. (p. 123) [14]. Based on these findings, Wikidata can be considered a trustworthy source for the language metadata required in this project.

The data sought includes all languages spoken worldwide, the countries and regions in which they are spoken, the number of speakers for each language, and the language family to which each language belongs.

2.1.2 OpenStreetMap and Sourcing Geographical Polygons

Visually displaying geographical areas on a map is done using polygons [15]. In geographic information systems (GIS), polygons represent specific areas such as vegetated areas, urban areas, and water bodies [15]. A polygon is defined by a series of coordinate pairs in latitude and longitude where lines are drawn between the coordinates that outline the boundaries of an area [16].

Border data for countries or regions are not available in the Wikidata database, therefore other sources needed to be considered when obtaining geographical data. For this project geographical data is sourced from OpenStreetMap (OSM) [17]. OpenStreetMap is a collaborative platform hosted by the OpenStreetMap Foundation, a non-profit organization dedicated to providing free and easily accessible geographical data and world maps [17]. Like Wikidata, OpenStreetMap is a crowd-sourced project where users contribute data to make geographical information available to everyone [17].

OpenStreetMap provides an API called Nominatim, which allows users to directly query the OSM database through code [18]. An API consists of protocols that allow software applications to communicate and exchange data with each other [19]. Each geographical element in the OSM database is assigned a unique OSM ID number, which can be used to fetch specific data [20]. Since the project is non-profit, it has limited computational resources, so there are strict API usage policies to prevent overloading their servers when retrieving large volumes of data [21].

2.1.3 Fetching Language Metadata

The Hypertext Transfer Protocol (HTTP) is a communication protocol widely used for web browser to web server communication and machine to machine communication. The protocol can also be used to define web-based API's [22]. HTTP defines several request methods, often referred to as HTTP verbs, which specify the intended action to be performed when communicating with an API. Common methods include GET for retrieving data, POST for submitting new data, and DELETE for removing existing data [23]. The digital locations where the API receives requests are the API endpoints [24]. This often consists of URLs, colloquially known as web addresses [25].

Language data from Wikidata is fetched by sending an HTTP GET request to the API endpoint provided by Wikidata: `https://query.wikidata.org/sparql`. This request includes a custom SPARQL query that specifies the needed data. Wikidata stores information as a graph database, where data is represented as interconnected entities, and the query language SPARQL is specifically designed to search through graph-based data [26]. The server returns the results in the requested format, JSON in this case.

To extract the data, two SPARQL queries had to be used due to the API timing out when attempting to get all the data simultaneously. The first query (see Appendix A.1) retrieved all entities with an ISO 639-3 language code and associated metadata, including the number of speakers, immediate language family, and language type. The ISO 639-3 code standard, established by the organization SIL Global, provides a set of three-letter codes that uniquely identify all known human languages [27]. To be eligible for a code, a language must be used by a group of people for human communication and have a documented history of use over time [27].

Fetching based on the presence of a ISO 639-3 code rather than querying all entities tagged as "Language" was a deliberate decision to ensure a comprehensible data collection. Wikidata has a number of different descriptor tags for what constitutes a language in use; some examples include "Language", "Modern Language", "Natural Language", to name a few. Due to the decentralized nature of Wikidata's moderation, there is a lack of standardization in how languages are categorized and labeled, thus returning inconsistent querying results. An attempt was made with querying all languages based of the descriptor tags, and six thousand out of

the roughly seven thousand languages were returned, thus going by ISO codes was more consistent in results. Further verifying that all "Languages", as defined per ISO 639-3, were correctly fetched and present in Wikidata, the official code table was cross-referenced with the pulled data [28].

The second query (see Appendix A.2) fetched the regions and countries where the languages are spoken. This includes whether the language is an official language of the countries found or not. The OSM ID number for each geographical region and country is also extracted to retrieve geographical polygon data when querying from the aforementioned API Nominatim[18].

2.1.4 Fetching Geographical Polygons

The Nominatim endpoint: `https://nominatim.openstreetmap.org/lookup?` takes one or multiple OSM IDs simultaneously, returning the requested polygons in a GeoJSON format [18]. GeoJSON is a file format based on the JSON file format designed to encode various geographical data structures, including polygons [29]. In GeoJSON, a polygon is represented by an array of linear rings, a linear ring being defined as an array of coordinates. The first linear ring represents the exterior boundary with optional subsequent rings representing holes in the polygon [29].

An optional threshold value can be specified when fetching from Nominatim, which simplifies the returned polygons [18]. The API utilizes the Ramer-Douglas-Peucker Algorithm, a graph reduction algorithm that reduces the number of points in a curve while minimizing the effect on the shape [30]. This was implemented to reduce the size of the polygon data, optimizing the required storage needed for the database of the language map. Setting the threshold value to 0.002, the size of the returned GeoJSON file is roughly 15 times smaller than the original size. By reducing the number of points in the polygons, the number of coordinates in the file is lowered drastically, minimizing the size of the polygon files while its shape remains visually identical. File sizes are further reduced another 15% by removing decimals of the coordinates, from seven to five, trading accuracy in favor of smaller file sizes [31].

2.1.5 Structuring and Cleaning Data

After each SPARQL query from Wikidata, the received data was structured into a custom JSON format fitting for subsequent handling. This, in turn, meant that the data could be stored between each handling process, helping with the continuous debugging process. The data was cleaned in successive stages, forming a data processing pipeline. First, all dead and constructed languages were sorted out. This decision was made for two main reasons: several of the regions historically associated with dead languages no longer exist, making it difficult or impossible to obtain their geographical region boundaries. Secondly, the project's scope is limited to currently spoken native languages with connections to one or several regions. Consequently,

constructed languages such as Esperanto were excluded, as their speakers do not originate from any specific geographic area. Lastly, retired ISO 639-3 codes were identified within the data and had to be removed as they no longer correspond to any actively spoken language [32]. To avoid unnecessary querying, this step was performed between the first and second SPARQL query.

The second step involved excluding dialects and sign languages, as these are beyond the scope of this project. Although excluded, the data for both dialects and sign languages are saved in separate files for possible future features. Several languages with no metadata were filtered out. Finally, the remaining regions with missing OSM IDs were cleaned and filled out by hand.

Once all regions were organized, the unique OSM IDs were run through the Nominatim API and fetched as described in 2.1.4. Returned data in GeoJSON format is further filtered, removing all irrelevant metadata from the response and solely saving the coordinate arrays.

2.1.6 MongoDB Atlas

To store and serve the processed data, the project used MongoDB Atlas, a cloud-based database service provided by MongoDB Inc [33]. MongoDB Atlas was chosen because it offers a flexible model that fits well with the JSON structure of the language and geographic data used in this project. Two collections are used, one for language metadata and one containing the coordinate data for the region border polygons.

By hosting the database in the cloud, the project could ensure consistent access across different development environments and securely connect it to the backend with the Mongo connection string stored in environment variables.

2.2 Backend

This section outlines the framework tools used to design and implement the backend for the interactive language map. The backend was designed to deliver structured language and geographic data to the frontend. It serves as the communication layer between the user interface and the database. When the user searches for a language or clicks on one in the interface, the backend receives the request, retrieves the appropriate information from the database, processes it into the correct format, and returns it to the frontend.

The backend was implemented using the NestJS framework version 11, and Node.js version 20.11.1 [34], [35]. NestJS is designed to support scalable and maintainable server-side applications with Node.js. NestJS was chosen for its modular and organized structure, which supports a clean code architecture and future scalability [34].

2.2.1 Request Handling and Endpoints

In NestJS, the part of the backend that receives incoming requests is called a controller [36]. A controller defines endpoints, which are specific web addresses that the frontend can use to ask for data. When the frontend sends a request to an endpoint, the controller receives it and passes it along to the responsible functions for processing and returning the data.

Each controller in NestJS is defined as a class, and each endpoint is represented by a method in that class [36]. The controller decides what kind of request it accepts and how the request parameters should be interpreted.

In this project, the backend used a controller to define three `GET` endpoints:

- `/language/all-names` to return a list of all language names
- `/language/geojson/:language` to return geographic information for a specific language
- `/language/country-details/:name` to return a list of all languages spoken in a specific country

The first endpoint, `/language/all-names`, was developed to return an alphabetically sorted list of all languages stored in the database.

The second endpoint, `/language/geojson/:language`, accepts the name of a language as an URL parameter. For example, `/language/geojson/swedish` would show areas where Swedish is spoken. The endpoint returns the full geographic distribution of that language in a GeoJSON format of type `FeatureCollection`.

A `FeatureCollection` groups multiple geographic features into a single object [37]. Each feature represents a country or region where the selected language is spoken. The features can store the area's shape, in this project as a polygon, using coordinate arrays. Each feature contains properties with additional information such as the region name, ISO 639-3 language code, official status, number of speakers, and the language family. This structure allowed the frontend to display relevant countries and regions for the selected language along with the metadata, in a single and organized response.

GeoJSON was chosen as the output format for several reasons. Firstly, it is supported by the frontend's mapping library, which allows direct rendering of spatial data in the browser.[37]. Secondly, GeoJSON integrates well with the MongoDB database, which supports geospatial queries in GeoJSON format [38]. Additionally, the format's simplicity and human readability make it easier to debug and manually inspect geographic data during development. These benefits made GeoJSON a suitable and practical choice for the system architecture and development process.

The third endpoint, `/language/country-details/:name`, returns a list of all languages spoken in a specific country. The name of the country is passed as an URL parameter and the backend returns a list of language names.

2.2.2 Data Processing and GeoJSON Construction

When the frontend requests geographic data for a specific language, the backend follows a structured, multi-step process to collect and prepare the response. The goal was to transform raw entries stored in the database into structured GeoJSON that the frontend map could display.

The first step was to search for the selected language in the database. To allow flexibility in user input, which is used for the search, the backend performs a case-insensitive search method for the requested language. To achieve this, the backend uses a regular expression query. A regular expression allows the system to match the language name exactly while ignoring differences with different letter casing [39]. For example, "Swedish", "swedish", and "SWEDISH" will all return the same result. This approach makes the system more forgiving of variations in input.

Once the correct language entry is found, the backend validates and processes the metadata. Data about the number of speakers is cleaned by removing entries marked as "missing" and converting numerical strings to integers for a consistent view. If available, the year the number of speakers was recorded is also included.

In the next step, the system iterates through all countries that are associated with the language. Each country is linked to an OpenStreetMap identifier, OSM ID, which is used to search for polygon data in the database. If no corresponding polygon data is available for the language, but only coordinate arrays, these are converted into valid GeoJSON polygon structures. Each country is then represented as a GeoJSON `FeatureCollection`, containing both the geographic shape and the associated metadata such as the country name, language name, speaker numbers, official status, and language family.

The same process was applied for the regions associated with the language. To avoid duplication, regions that are already listed as countries are excluded. Each region includes the same metadata as for countries, along with the associated country and an indicator that the language does not hold official status in that specific area.

If no valid country or region polygons can be found, the system returns an error response indicating that no usable geographic data are available. If only some entries are missing, the system processes the remaining valid values.

The resulting GeoJSON structure allows the frontend to directly visualize the geographic distribution of the selected language, along with the associated information.

2.2.3 Data Structure

As part of the backend implementation, a structured approach was taken to ensure consistency in how language data is stored and processed. Although the data in the database is stored in JSON-like format, there are no rules for what each entry must include. To avoid problems with inconsistent or incomplete data, the backend defines a clear structure for how language entries must be stored and what information they should contain. This structure is often referred to as a schema in NestJS, and acts as a set of rules that describe the expected format and contents of each document in the database [40].

The backend applies this structure using a library called Mongoose [40]. Mongoose allows the system to specify which fields are required, what value they should hold, and how arrays of countries should be organized. By doing this, the system can validate the data automatically when communicating with the database, which helps prevent errors and simplifies the logic needed to process the data [40].

Each language entry followed a consistent format, which included:

- **Language Name and ISO Code:** Basic identifiers used in search results and metadata responses.
- **Countries:** An array of objects, each with a country name, one or more OpenStreetMap identifiers for retrieving geographic shapes, and a flag indicating whether the language had official status in that country.
- **Regions:** An array of sub-national areas where the language was spoken, each with a name and identifier.
- **Speaker data:** Optional entries describing the number of speakers and when that number was recorded.
- **Language family:** One or more categories describing how the language is related to others.

This structure ensures that each language has complete and consistent associated metadata, ready to be processed when the data is transformed into GeoJSON format for the language in the frontend.

2.2.4 Error Handling and System Configuration

Robust input validation and error handling were necessary to maintain reliability and prevent incorrect or incomplete data from affecting the system. To support this, several mechanisms were implemented in the backend to detect invalid inputs and respond appropriately. For example, if a user searches for a language that does not exist in the database, the system returns a response indicating it was not found. This prevents the frontend from receiving undefined or misleading data.

The backend also accounts for cases where some countries or regions do not have the corresponding geographic shapes. If one or more entries are missing polygon data, the system logs a warning internally but continues to process the remaining entries. This ensures that users can still receive a result rather than an empty response. However, if no valid polygons are found, the system returns a not found error to indicate that no map could be generated for that language.

The application establishes a connection to the MongoDB database using Mongoose's asynchronous configuration method[1], allowing specification of details such as the MongoDB connection string [40]. The connection is securely fetched from environment variables at runtime, ensuring that sensitive information is kept out of the public codebase and enabling flexibility.

[1]Full identifier: `MongooseModule.forRootAsync`

2.3 Frontend

The frontend of the application was developed using React and TypeScript, with Vite as the build tool and Render for deployment. This section describes the tools, architectural structure, and design patterns used to build an interactive language map with an user-friendly interface.

2.3.1 Environment and Frameworks

The frontend was initialized using a public Vite-React-TypeScript template[2], which offered a good structure for expanding development [41]. The template supported the use of React and Typescript as the primary frontend development tools. Vite was chosen as the build tool for its fast startup and efficient module handling for fast development [42].

React is a JavaScript library used for building user interfaces for web applications. React was chosen as the frontend framework for its component-based architecture that allows developers to create reusable user interface (UI) elements [43]. TypeScript was adopted to enhance reliability through strong type checking, which means that the data values needed to be explicitly marked as a specific data type [44]. Since TypeScript converts to JavaScript, it is possible to use TypeScript in all ways that JavaScript could be used for [44].

Render was later used to provide reliable and free web hosting for both the frontend and backend application. Render is a cloud platform that supports continuous deployment from a Git repository, automatically rebuilding and publishing the frontend whenever changes are pushed [45]. This allowed the application built with React, TypeScript, and Vite, to be deployed as a dynamic, production-ready web service. The project was organized into modular folders separating pages, reusable components, styling, and utility functions, promoting a clean and scalable architecture.

2.3.2 Component and Page Structure

To maintain modularity and the ability to reuse code, the frontend was structured using a component-based approach. The `components` directory contains reusable elements, categorized into sub-folders based on their usage, such as `globalComponents`, `homeComponents`, and `worldMapComponents`.
In addition, two global components were implemented, a navigation bar and a footer. The main objective of using global components is to create a coherent experience

[2]Full name: `vite-template-react-ts-jest`

for the user when navigating the website, since the pages maintain familiarity with each other.

Each main page of the application, such as `Home`, `World Map`, `Credits`, was implemented as separate React components within the `Pages` folder. This allows for each page to be developed and styled with its own CSS independently. This approach avoids global style conflicts and supports scalability.

To further upgrade the development process, more complex components were imported from Material UI (MUI), a component library for React [46]. MUI was used for certain components such as buttons and text fields, to reduce development time and ensure design consistency across pages. Components not available in MUI were implemented manually.

2.3.3 Interactive Map Features

The interactive map was encapsulated within a single React functional component, `Map.tsx`. When this component loads, it creates a base-map using MapLibre, an open-source mapping library for rendering interactive maps with customizable styling and high performance [47]. The map uses a light background so that the colored language areas stands out clearly.

A `useEffect` hook was implemented to track changes to the `selectedLanguages` array, which is provided via React Context, a way to pass data down to the children of the components without the need to pass through every child component [48]. This hook compares the newly selected languages to those already loaded on the map. For each new selection of languages, the component sends a request to the server to fetch GeoJSON data. Upon receiving the data, the component ensures each feature contains a unique numeric ID. This unique ID is essential for managing the feature state via MapLibre's feature-state API. Feature-state means that it can dynamically update the map visuals based on user interactions without changing the original data. For instance, selecting a language could update its state as follows: "selected": true [49]. The GeoJSON data is then added as a new source, and two layers are created, one for filling the language areas and another for outlining their borders. The fill layer applies one color for official languages and a different color for unofficial languages. To improve interactivity, the opacity of the fill layer increases when hovering over a region or country. For the outline layer, the stroke color is determined by hashing the language name into an RGB value and applying a darkening function to ensure sufficient contrast between different selected languages.

Event listeners on the fill layer handles hover and click interactions. On hover, the opacity of the previously hovered region is reset and the new feature's opacity increases. When the cursor is leaving a region, the hover state is cleared. Clicking on a region opens a MapLibre popup at the click location. The popup's content is dynamically generated using template literals displaying the language's name,

the associated country (and region, if available), the language family, and the most recent number of speakers. A custom close button in the popup allows users to dismiss it.

If a language is later removed from the selection, the component cleans up by removing the fill and outline layers and closing any active popups related to that language. This automatic removal is handled via a cleanup process that ensured all layers and popups are properly disposed of.

2.3.4 Filtering and State Management

To manage visibility of geographic features depending on user preference, a `viewFilter` state was implemented via React's `useState<ViewFilter>` hook inside the `MapComponent`. This state is updated via a UI control component named `CountryCheckbox` and can take one of three values: "Default View", "Country" or "Region". The `CountryCheckbox` renders these options as radio buttons and includes a built-in legend indicating the color coding for official (green) versus spoken (yellow) languages, and provides an explanatory tooltip to help users understand how data will be shown in the "Default View" mode. The tooltip reads:

> Displays every territory where the language occurs:
> - Countries and regions where it's official
> - Regions if it's unofficial but has regional data
> - Whole countries if no regional breakdown exists

The actual filtering logic is handled through MapLibre's `setFilter` method, which applies small filter expressions to each layer corresponding to a selected language. These expressions determine feature visibility based on each GeoJSON feature's properties:

- `"Country"`: shows only country-level polygons (features *without* a `region` property).
- `"Region"`: shows only sub-national polygons (features *with* a `region` property).
- `"Default View"`: combines all features through a logical expression that displays
  1. all countries and regions where the language is official
  2. regions where the language is unofficial but region-specific data exists
  3. whole countries when no regional data is available for unofficial usage.

2.3.5 Interface Design Patterns

The interface design was guided by established user interface (UI) design patterns. [50]. These patterns help the user to feel similarity between different websites and are key to creating an easily navigated platform [50]. The following patterns, based on the categorizations from the website UI-Patterns.com [51], were applied because of its large library of different patterns that could be used:

- Clear Entry Points: When the user first logs into the website, they will see a large map and a button that tells them that they will go to the main feature of the website, the world map.
- Visual Framework: The website uses the same color scheme, font styling, and overall style on all its pages.
- Navigation tabs: Used at the top of the website to indicate the different pages the user can navigate to.
- Fat Footer: Used at the bottom of the website to let the user navigate the website more quickly. Its secondary usage is to display links to other media that could be used by us.
- Input prompt: Used in the input field used to search for a specific language on the map page.
- Continuous scrolling: Used in the language button container.
- Cards: Used on the credits page to easier display repeated information about each and every member that has contributed.

2.3.6 Hosting Using Render

Render was chosen as the hosting platform due to its simplicity and support for full-stack applications. Since the application requires both a frontend and backend to function together, each was deployed separately on Render as individual services. The frontend, built with TypeScript and Vite, was configured to communicate with the backend via API calls over HTTPS. This is the same as HTTP with the only difference that it is encrypted. [52]

2.4 User Tests

To evaluate the functionality and usability of the product, a structured user test was conducted. Usability testing is a widely recognized method in human-computer interaction, focused on observing real users as they perform meaningful tasks to identify potential design issues [53].

The test was designed as a small-scale study and included three key elements: task-based scenarios, a think-aloud protocol, and follow-up analysis which can provide valuable insights according to Barnum [53]. The user tests were conducted during the later stages of development, when a functional first version of the map was completed, but with a few weeks remaining for further improvements based on feedback.

A total of 10 participants were selected based on their alignment with the intended target user group. Each participant completed the test individually. The evaluation aimed to determine whether users could intuitively navigate the interface, search for specific languages, locate their geographical distribution, and interpret related metadata without prior guidance.

Participants were asked to complete three predefined tasks:

- Search for a specific language and identify its geographic distribution
- Find the number of speakers for the specific language
- Exploring the interface freely to assess navigation and content

Throughout the test, participants were encouraged to verbalize their thoughts using a think-aloud process. This method provided insights into their interactions, as well as the reasoning behind their decisions and difficulties encountered [53].

Observations and verbal feedback were documented and used to identify usability issues. The feedback was used and evaluated for possible improvements.

2.5 Alternative Methods

Throughout the development of this project, several alternative approaches were considered and tested before the final implementation. This section outlines the methods that were evaluated. In each case, the alternatives were assessed based on their practicality, performance, and ability to represent language data accurately. The following subsections explain the reasoning behind the decisions to exclude certain methods in favor of other, more suitable solutions for the project.

2.5.1 Visualize Linguistic Presence

One of the challenges encountered during the project involved visualizing language distribution using data retrieved from Wikidata. When querying Wikidata for a specific language, the resulting data typically includes regions and countries where the language is spoken. However, this often leads to an overgeneralized representation. For example, when retrieving data for French, the system identifies France, Morocco, Algeria, and Canada, countries where French is widely spoken. However, the data set also includes countries such as India, the United States, and Italy, where French is spoken only in limited contexts and not nationally.

This raises the question of how to define and visualize the linguistic presence in a way that accurately reflects actual usage without misleading the user. Although it is technically correct to include all countries where a language is spoken, representing them equally on a map may give the false impression that the language is widely spoken throughout those entire countries. This can result in a misinterpretation of the linguistic landscape.

To address this, one approach considered was displaying only countries where the language has official status on the map, while listing other countries in a supplementary section. Alternatively, a categorization system was explored in which different shades, colors, or patterns would represent varying levels of linguistic presence.

In the final implementation, a color-coding system was applied to represent the different levels of linguistic presence between countries and regions. Rather than showing all countries equally, the map now uses green to highlight countries where the language has official status, and yellow to represent areas where the language is used only regionally or unofficially.

2.5.2 Database

Another challenge in this project was deciding how to manage and store the language data retrieved from Wikidata. A key decision was whether the system should dynamically fetch data from Wikidata and generate a GeoJSON file on request, or store the data in a dedicated database and serve it from there.

The initial implementation used a dynamic approach, querying data from Wikidata in real time. While this ensured access to the most up-to-date information, it quickly led to performance issues. Processing the data dynamically caused noticeable delays, especially when handling complex or large datasets. In addition, any disruption in communication with external databases would make the language map unusable. As a result, this approach proved unsuitable for the intended use.

A hybrid model was also considered, in which language-related data would be stored in a local database, while geographical coordinates for generating GeoJSON files would be retrieved dynamically from OpenStreetMap. This would lessen delays but run into the same communication-related problems as stated above.

The final solution involved storing all relevant data, including both language information and geographic coordinates, in the cloud database MongoDB Atlas. This approach ensured faster response times and provided greater control over the structure and integrity of the data. Because the data is stored independently, temporary issues with the external database do not affect the availability of the language map.

3

Results

This chapter presents the results of the system implementation and the user interface of the language map. The source code of the implementation is available on GitHub through the following link: https://github.com/hannamagn/InteractiveLanguageMap2

3.1 Database

The backend fetches information from the MongoDB Atlas database on demand after data from Wikidata and OSM has been fetched and cleaned.

3.1.1 The Data Pipeline

The preprocessing method of the data described in 2.1 is centralized in a separate python script known as the Database Collector. Figure 3.1 shows a model of the data pipeline. Running the script automatically re-queries all data and either updates or adds new data to the database. While automatic, running the script has to be done manually, resulting in possibly outdated information if not run regularly. During the filtering process, a total of 358 dialects and 158 sign languages were sorted out.

3.1.2 Contents and Structure of the Database

Data for this project is stored in two collections: one collection containing the language metadata as well as the regions and countries where this language is spoken, and another collection that stores all the geographical polygon data of the various geographical places. Currently, there are 6741 languages recorded together with 2386 unique geographical regions stored in the database. Any data not present in the Wikidata database is marked as "Missing". The structure of these collections is shown in the figures below. Data in these collections is stored in BSON format, a binary JSON format that lends itself to easily be converted to GeoJSON, the data structure used in the frontend to show the information on a world map [54].

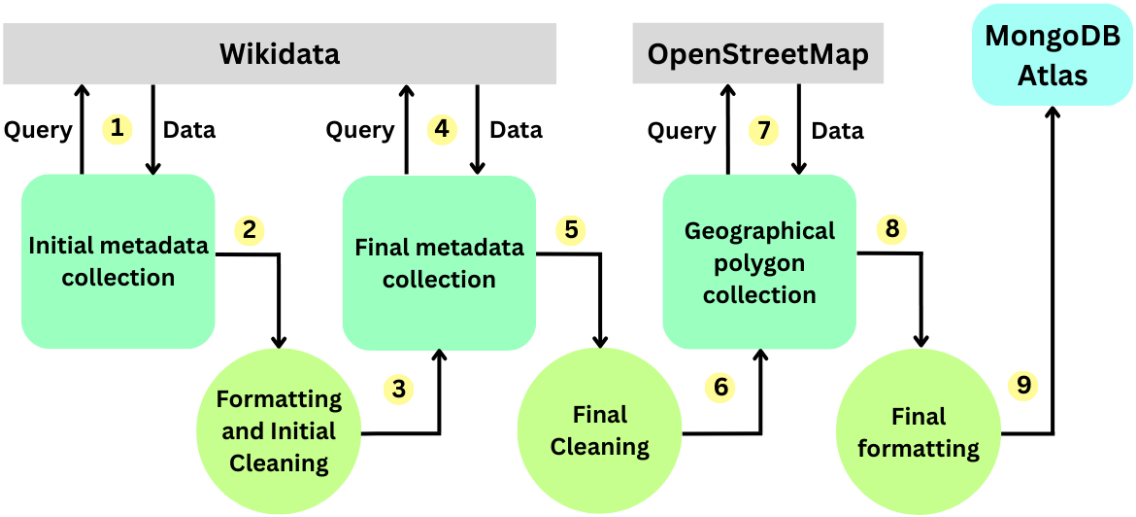


Figure 3.1: The data pipeline with numbered 1-9, corresponding to the order of execution.

3.2 Backend

The backend was implemented using the NestJS framework and served as the communication layer between the MongoDB database and the frontend. It handles HTTP requests for language data and returns structured JSON and GeoJSON responses to the frontend.

3.2.1 Endpoints and Integration

The three backend endpoints, `/language/all-names`, `/language/geojson/:language`, and `/language/by-region/:name` were fully implemented and returned the expected data formats for the language map in the frontend. All endpoints were individually tested and worked as expected. Here is a breakdown of their use:

- The `/language/all-names` endpoint returns an alphabetically sorted list of all language names. In the frontend, this list populates the language search box, enabling users to find and select languages dynamically with search filtering for performance optimization.
- The `/language/geojson/:language` endpoint returns geographic and metadata information for a specific language in valid GeoJSON format. This is dynamically loaded into the map component in the frontend when a language is selected, allowing for its visual representation.
- The `/language/country-details/:name` endpoint retrieves language details for a given country. It returns a structured list of both official and

Figure 3.2: Example structure of the language metadata collection (left) and the polygon data collection (right) as shown on the MongoDB Atlas website.

non-official languages associated with that country, enabling the system to distinguish between their statuses. The data is shown in the frontend in a styled popup when user clicks on a country in the map.

The following two examples illustrate the structure and responses of the two primary endpoints, showing how language data is retrieved and formatted for frontend integration.

1. Retrieval of all language names
Request:

```
GET /language/all-names
Accept: application/json
```

Response (200 OK):

```
[
"Are'are",
"Auhelawa",
"Ole",
"A-Hmao",
"A'ou",
...
]
```

2. Retrieval of GeoJson for a specific language
Request:

```
GET /language/geojson/German
Accept: application/geojson
```

Response (200 OK):


```
{
  "type": "FeatureCollection",
  "properties": {
    "language": "German",
    "iso_code": "deu",
    "countries": [
      "Liechtenstein", "Czech Republic", "Germany", "Russia",
      "France", "Austria", "Switzerland", "Italy",
      "Poland", "Denmark", "Luxembourg", "Belgium"
    ],
    "regions": [
      "Austria", "Germany", "Trentino-South Tyrol", "Trentino", ...
    ],
    "language_family": [
      "West Germanic", "South Germanic"
    ],
    "number_of_speakers": [
      {
        "number": 80000000,
        "appliesTo": "second language",
        "timeSurveyed": "2012-01-01T00:00:00Z"
      },
      {
        "number": 76540740,
        "appliesTo": "first language",
        "timeSurveyed": "2019-01-01T00:00:00Z"
      }
    ]
  },
  "features": [ ... ]
}
```


3.2.2 GeoJSON Output

The backend dynamically generates a GeoJSON `FeatureCollection` for each language, consisting of geographic polygons and associated metadata such as language name, ISO code, associated countries and regions, number of speakers, and language family.

In a few cases, polygon data was missing for certain regions or countries. When

this occurred, the system logged a warning, but continued processing valid entries. This allowed the frontend to display partial results rather than failing. The output remained compatible with the frontend's mapping library even in cases of incomplete data.

The backend supported case-insensitive language searches using regular expression queries. This allowed the users to enter language names in any capitalization and still receive accurate results.

Speaker numbers were also handled well. Entries marked as missing were excluded from the response, and valid numbers were cleaned and formatted consistently. The survey year was included when available.

3.3 Frontend

The frontend provides the user interface for interacting with the application. It is structured into three main pages: Home, World Map, and Credits. The pages are designed for a clean and intuitive look, and the color schema is consistent throughout the website.

3.3.1 Global Components

Global components are used on all or many pages on the website. There are two global components: a top navigation bar and a footer. The global components are used to ensure that the website is coherent throughout all of its pages and to let the users have a feeling of familiarity between pages.

Figure 3.3 shows the navigation bar that allows users to switch between website pages. The currently active page is visually indicated in grey to support interactivity.

Figure 3.3: The navigation bar is always shown on the website, allowing users to access each page.

As seen in Figure 3.4, the footer provides additional links and contributes to a consistent user interface across the website.

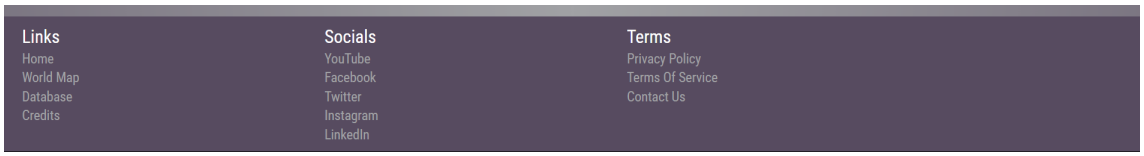


Figure 3.4: The footer is always shown on the home page and credits page. It provides additional links for user navigation.

3.3.2 Home Page

The home page, shown in Figure 3.5, serves as the welcome page for users upon accessing the website. On the home page, the users are introduced to the project and its purpose, as well as the possibility of familiarizing themselves with the website interface. It uses a clear visual entry point to direct the user to the world map page.

Figure 3.5: The home page of the website, showing the clear entry point for navigating to the world map.

When hovering on the entry point to the world map, the button becomes highlighted to indicate interactivity. This is seen in Figure 3.6.

Figure 3.6: The clear entry point gets highlighted on hover to visually show interactivity.

Information about the project is shown on the home page when scrolling down, as seen in Figure 3.7.

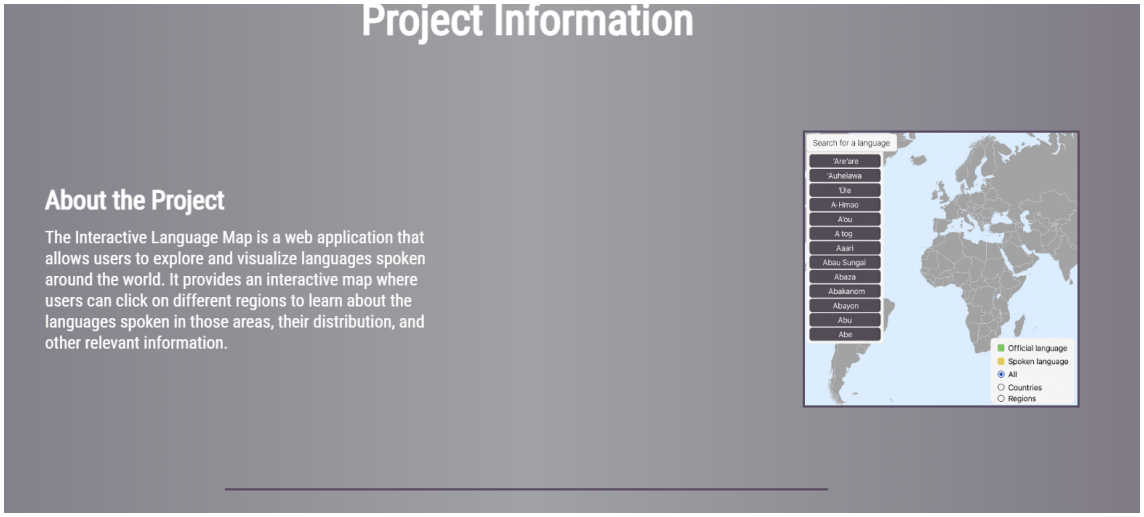


Figure 3.7: Information about the project.

3.3.3 World Map Page

The world map page is the main feature of the website. At first, a plain world map is shown, a searchable list with languages, and filtering functionality. The first view of the world map is seen in Figure 3.8.

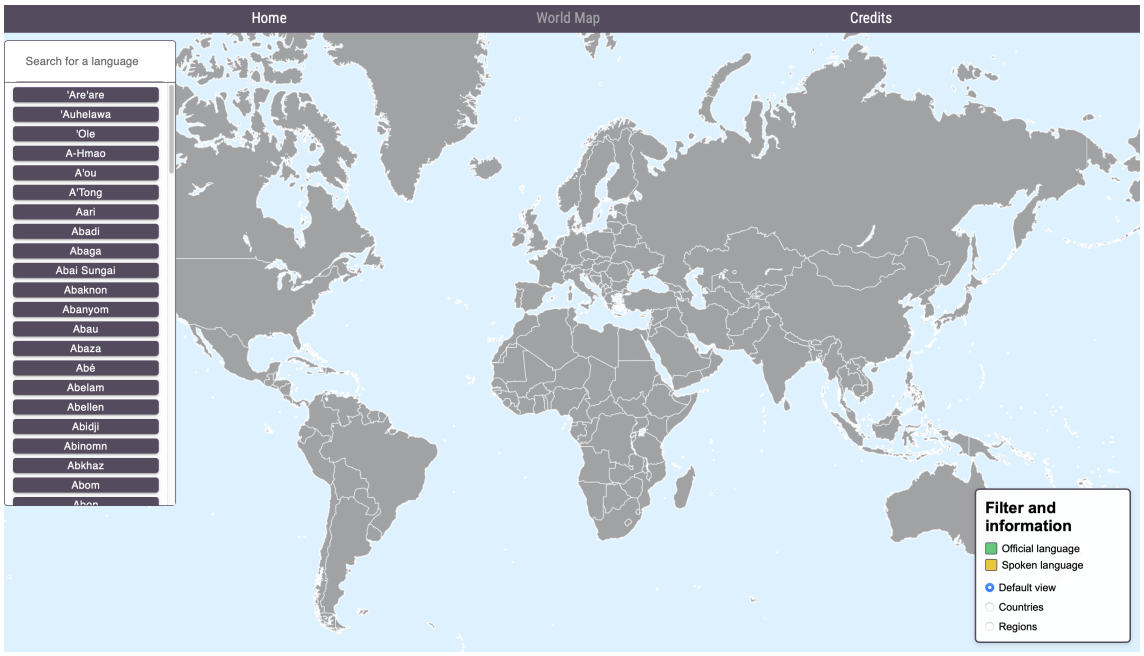


Figure 3.8: The start page of the world map, showing a global view with selectable countries, a searchable language list on the left, and filter options on the bottom right.

A list with languages and search functionality is displayed on the left of the page, as shown in Figure 3.9. Initially, only 100 languages are loaded in the list, with more appearing as the user scrolls. A search field is available to find a specific language quickly.

Figure 3.9: The search box that allows the user to search and scroll in the list for a language. The list contains all spoken languages.

In the lower right corner, there is a box with functionality for users to filter the view by default, regions, or countries. The filter box is seen in Figure 3.10. A small legend explains the meaning of the colors for different areas, distinguishing the languages' official or non-official status.

Figure 3.10: Filter and information box for selecting different views on the map and informing the user about the colors used in the interface.

When hovering over the label, a tooltip appears explaining the default view. As shown in Figure 3.11, the tooltip states that the default view displays countries and regions where the language is official and regions where the language is non-official. If no regional data are available but the language exists in the country, the entire country is shown.

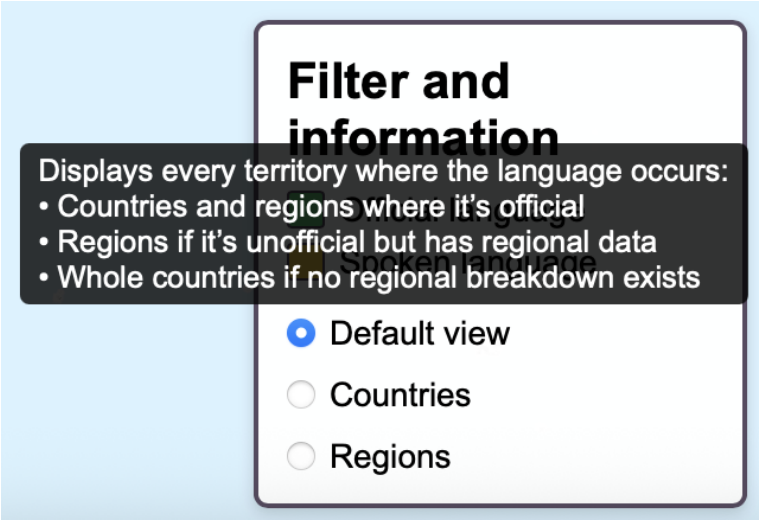


Figure 3.11: When hovering on the label "Default view", a tooltip displays with information about what it means with default view.

When a language is selected in the list, its button changes color to indicate selection and appears in a separate container at the top. The separate container makes it easier to manage and remove the selected languages. This is seen in Figure 3.12 where French is the selected language, and the button changed color and is added to the container at the top.

Selected languages are visualized on the map using color-coded areas. Green areas indicate where the language has official status, while yellow areas indicate non-official areas. Each of the selected languages has a uniquely colored outline to distinguish multiple languages if selected, and the unique color matches the outline on the associated language box.

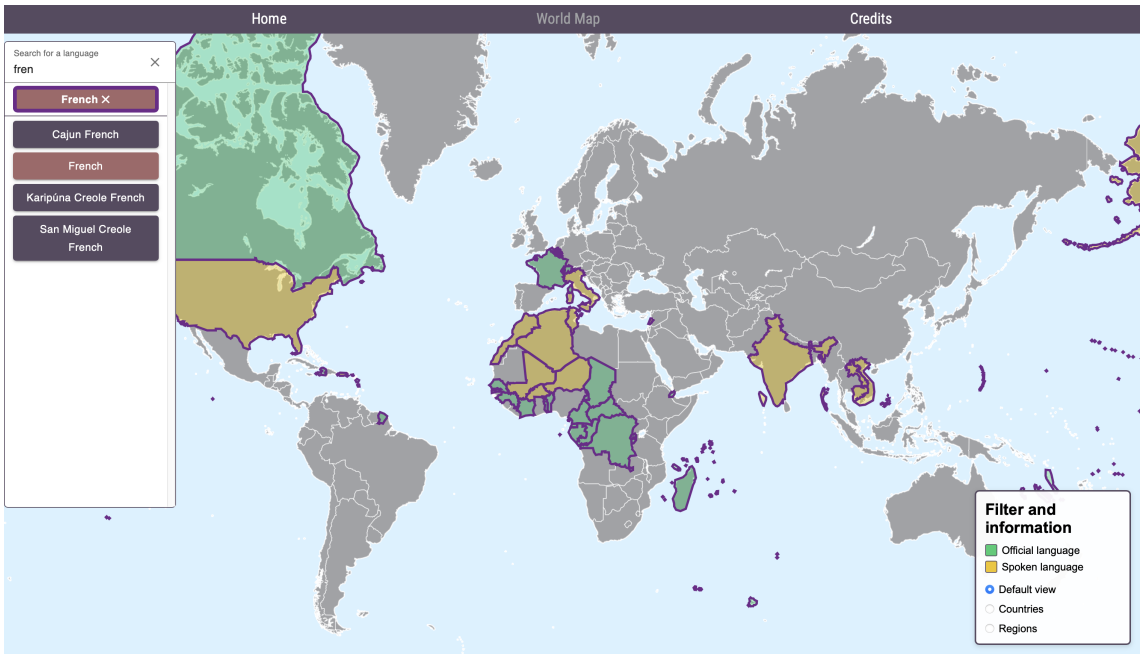


Figure 3.12: French is searched for and selected in the list. The map shows all countries and regions where French is an official or spoken language.

Hovering over a region increases its opacity to highlight interactivity. Clicking on a country or region opens a popup box with detailed information about the language, as seen in Figure 3.13. The popup contains information such as the name of the language, country, region, language family, and number of speakers.

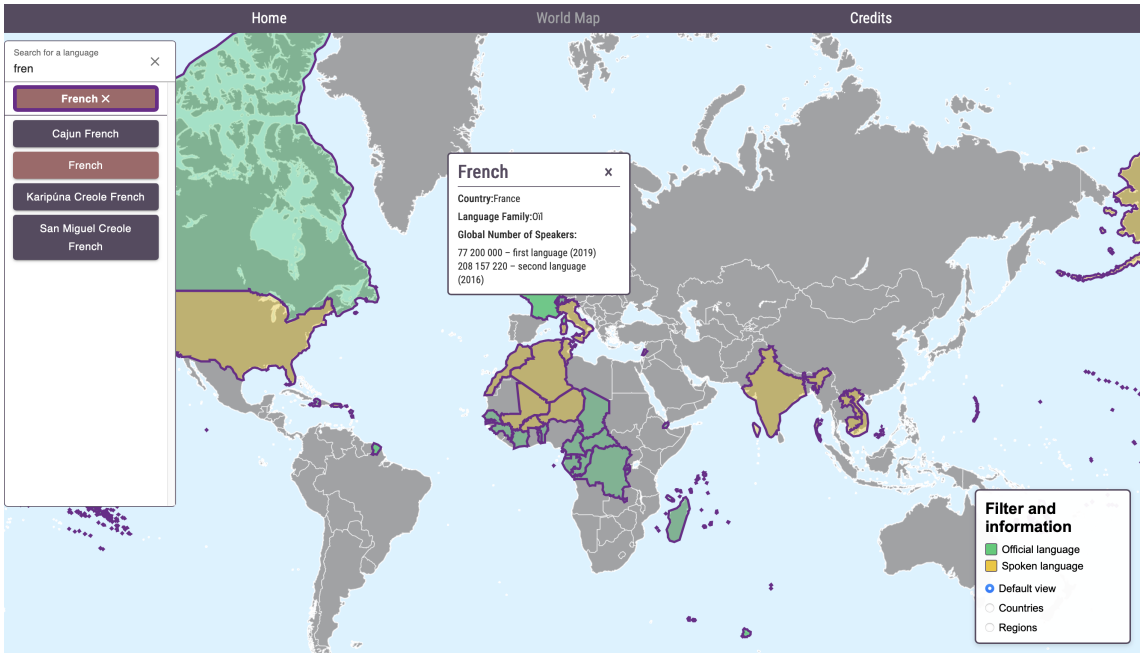


Figure 3.13: When clicking on a displayed country for French, a popup shows associated metadata.

Figure 3.14 shows a detailed view of the popup when clicking on France. Information about region is not shown in this popup, since France is a country. If the number of speakers has many recordings, the popup only shows the latest one.

Figure 3.14: Detailed view of the popup displayed.

Figure 3.15 presents a detailed version of the default view for French. Belgium and Switzerland are displayed both with the whole countries, but also with specific regions. The countries are shown due to French being an official language at the national level.

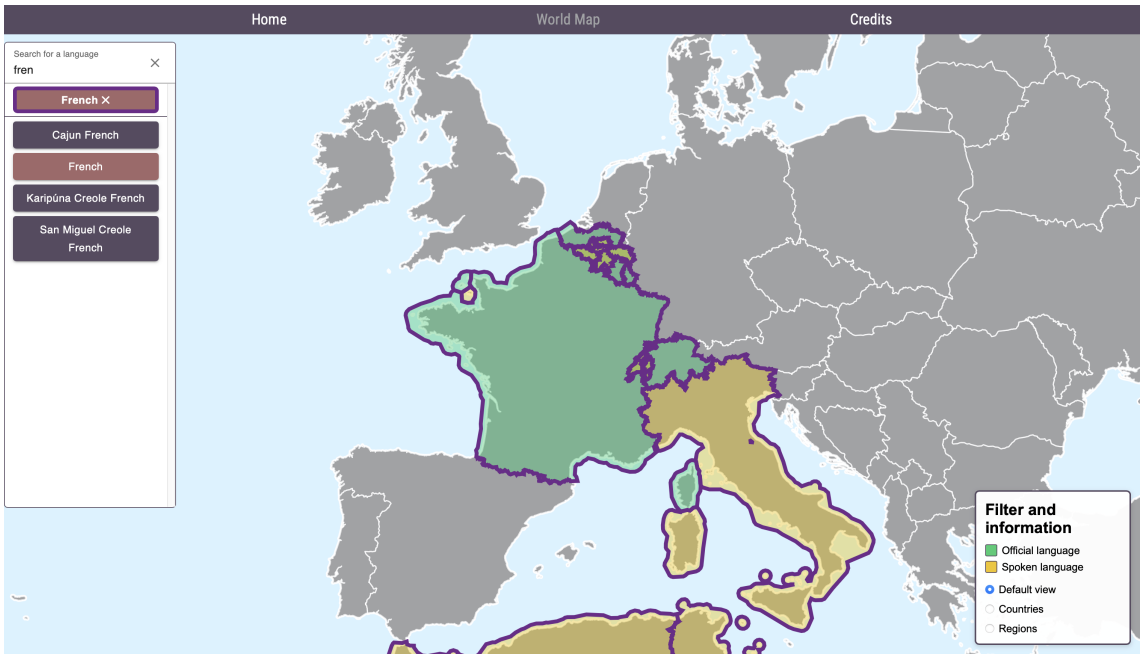


Figure 3.15: A zoomed-in version of the default view for French.

Figure 3.16 presents the map when the countries view is selected in the filter box. This view displays all countries where French is spoken.

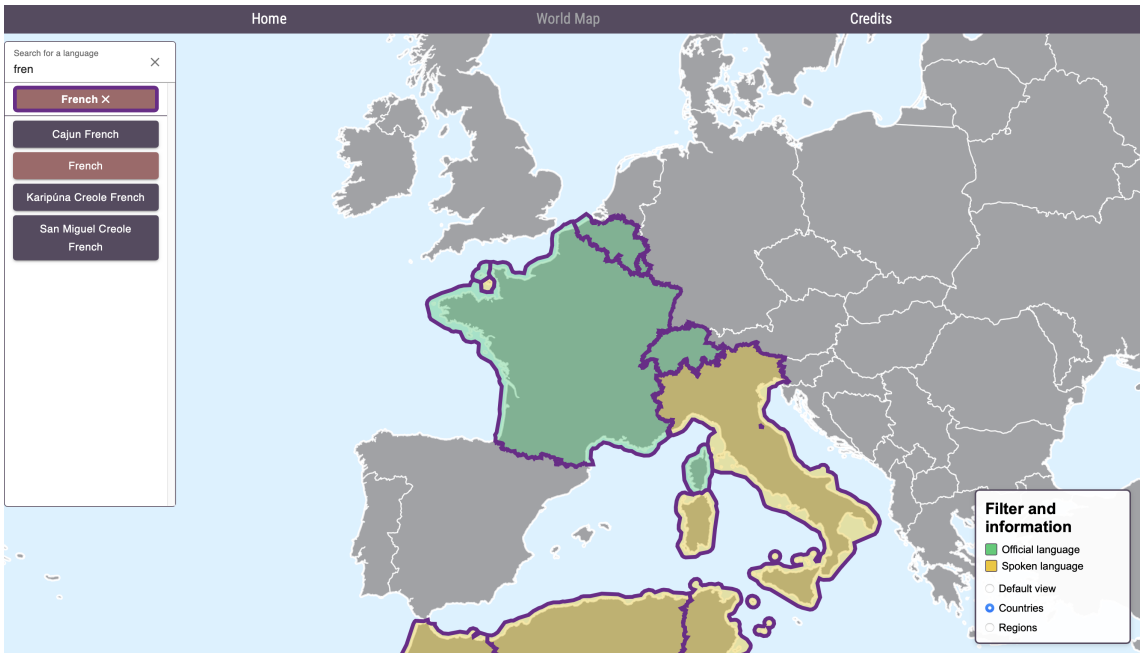


Figure 3.16: A zoomed picture of the view with only countries for French.

When the regions view is selected in the filter box, only the regions associated with the language are shown, as seen in Figure 3.17.

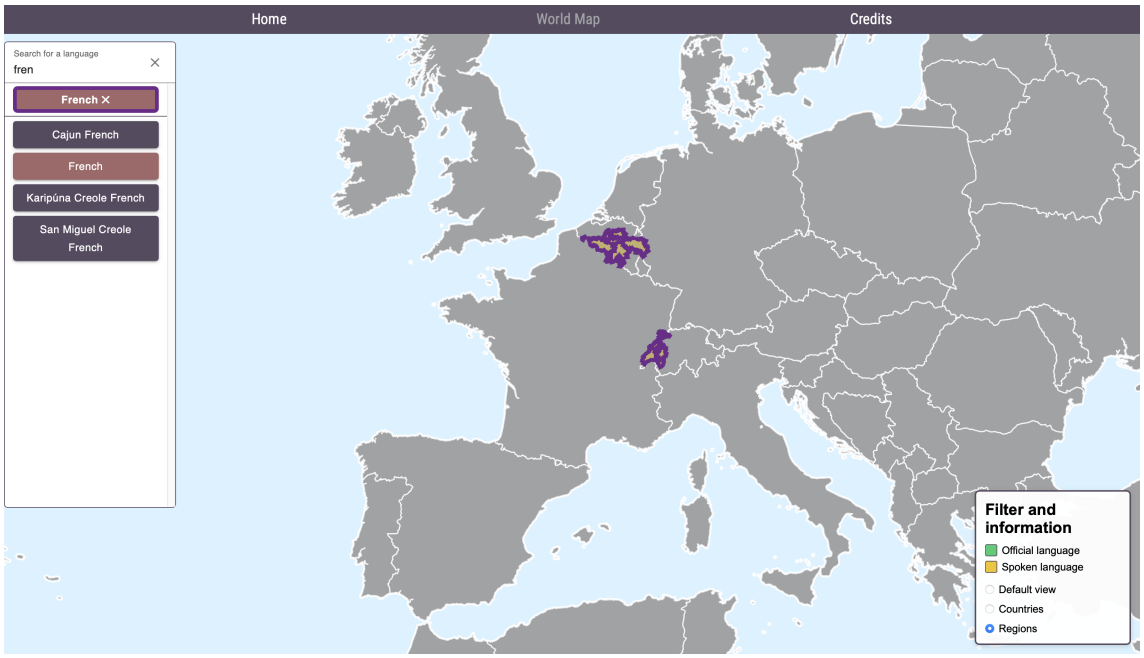


Figure 3.17: A zoomed picture of the view with only regions for French.

Figure 3.18 shows an example with multiple selected languages, each assigned a unique color used for both map outlines and selection buttons, making it easy to distinguish languages and their regions.

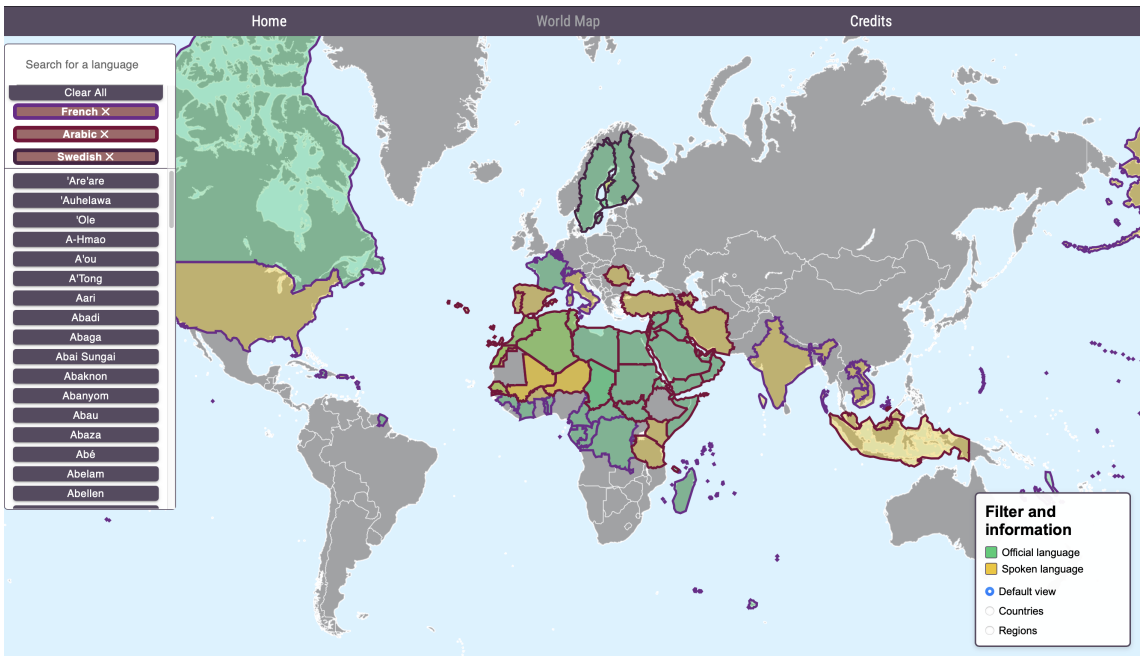


Figure 3.18: Three languages are selected in the list, showing the matching colored outlines for clarity.

When clicking on a country on the map, a popup will appear that displays both the official language in that country, but also other non-official languages that are spoken, as shown in figure 3.19 and 3.20.

Figure 3.19: Popup showing the spoken language when clicking on a country.

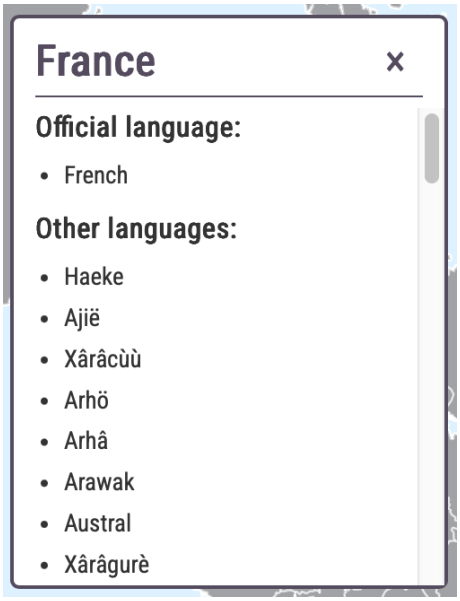


Figure 3.20: Zoomed-in view of the popup displaying the spoken language.

3.3.4 Credits Page

The credits page showcase the team members who contributed the development of the project. Figure 3.21 shows the credits page where each person is featured on an individual card with a description of their contributions.

Figure 3.21: The credits page showing the team members who contributed to the project.

3.4 User Test Results

The results from the usability test indicated that most participants could complete the assigned tasks without guidance. The interface was perceived as intuitive, and users generally found it easy to search for a language and understand the information displayed. However, the test also highlighted several usability issues.

Many participants first clicked on a country to directly interact with the map, which did not work at the time for the user tests. They felt like the map lacked some interactive features, which resulted in the inability to explore the world map in an intuitive way.

Another area that was acknowledged was the accessibility of the website, particularly for those with loss of color vision. The participants noted difficulty in distinguishing between different regions due to low color contrasts.

Some participants found the pop-up box with metadata difficult to scan. Suggestions about improvements included increasing the font and making labels, such as "Country" and "Language" bold to make it clearer. Some participants also misunderstood the speaker data, where the label was "Number of Speakers", assuming it represented the number of speakers in the selected country or region, rather than globally.

Finally, participants pointed out a lack of consistency between the selected language button and the map colors. They noted that if the outline of the selected language box matched the outline of the represented areas, it would be clearer how they align.

Based on the feedback collected from the user tests, improvements were implemented for the final version of the product. Interactions for clicking on countries directly on the map were implemented, enhancing the encouragement to explore the map. The outlines for each area on the map, matches the outline of the selected language in the search box. The pop-up box was improved with clearer labels along with larger and bold fonts. These improvements enhanced the user experience and ensured that the interface of the language map met the users' needs.

4

Discussion

This project's purpose was to develop an interactive language map that visualizes the global distribution of spoken languages. The aim was to address the lack of high-quality, customizable, and interactive tools that allowed users to search for individual languages and visualize their geographical distribution and associated metadata.

The implemented language map aligns well with the defined purpose and objectives. It enables users to search for individual languages, see where in the world they are spoken, and access metadata such as the number of speakers and official status. The application therefore addresses the identified gap in the availability of high-quality and interactive language maps.

Feedback from user tests further supports this alignment, as participants were generally able to complete the tasks given. Although we managed to get many different views on how we could elevate our product further using user tests, some things we could have done better would be to test a wider variety of users, or we could have done more user tests to get a better understanding of where we fell short.

4.1 Limitations and Accuracy

An aspect of the aim of this project was to address the lack of high-quality language maps. Accuracy is a vital aspect of quality, so an analysis of accuracy is warranted, as well as limitations that would lead to lesser quality.

4.1.1 Database Uncertainty

The created language map is based on two separate existing databases: Wikidata for the language metadata, and OpenStreetMap for the geography polygon data. These are both openly edited databases, which means they are prone to contain some unreliable data. Although some filtering was applied to remove dead or constructed languages, geographic or speaker information accuracy may vary.

Due to the limitations of the project to these two databases, the accuracy of the map is not comprehensively analyzed. Currently, the true accuracy of the map can only be described as mirroring the accuracy of Wikidata and OpenStreetMap.

4.1.2 Polygon Complexity and Memory Optimization

During development, rendering large and detailed geographic polygons posed significant performance challenges. While rich in data, these shapes often led to high memory usage and occasional crashes, particularly when attempting to load data for multiple languages at once.

To address this, simplification techniques were applied as mentioned in Section 2.1.4, to reduce the complexity of the polygon shapes. Even if this ensured smoother performance and faster loading times, it introduced a trade-off between visual accuracy and system efficiency. Preserving too much detail slowed the application, while overly simplified shapes risked misrepresenting geographic areas on the map. Finding the right balance was done by comparing different levels of simplifications between two polygons bordering each other. If the borders started overlapping, and that overlap was visible from the applications when zoomed in to the max, the shapes were deemed oversimplified.

Another aspect was dealing with the varying levels of detail between the fetched polygons. A number of polygons consisted of only a few points to begin with, and applying more simplification on top of that transformed them into squares or triangles. That meant some very detailed and large polygons could have been further simplified without issues, but were not since the same simplification threshold (see Section 2.1.4) is used for all polygons.

This experience highlighted the importance of thoughtful data optimization in building scalable, interactive mapping tools, especially when dealing with large datasets that must remain visually intuitive and responsive in a browser environment.

4.1.3 Use of GeoJSON

Using GeoJSON was a practical choice during development due to its integration with the frontend and the backend, MapLibre and MongoDB. During development, the simplicity and readability of GeoJSON also facilitated manual debugging and editing, which proved helpful when styling layers or testing data interactions. Although we considered using KML, an XML-based format designed for geographic visualization, supported by GIS applications such as Google Earth [55], as an alternative, its structure and more limited integration with MongoDB and MapLibre made it less suitable for our case.

Overall, GeoJSON offered the operability and flexibility required to bridge our

backend database and frontend map interface effectively. However, it has limitations, such as its lack of support for topology and 3D geometries, and relatively large file sizes for complex datasets, which may pose constraints in more advanced or performance-critical applications. Future extensions of our application could benefit from exploring alternatives like vector tiles, which enable dynamic styling and smooth interactions, or more efficient formats such as FlatGeobuf, which is designed for high performance and spatial indexing [56] [57].

4.2 Societal and Ethical Aspects

A mapping of the world's languages raises some potential ethical concerns. One issue is the potential misrepresentation or simplification of the linguistic reality, like establishing a precise language border when the border in actuality is vague and overlaps with other borders [58]. This ignores nuances and could potentially imply false divisions, which could lead to the marginalization of minority languages.

Creating a language map also have the potential of ethical benefits, like spreading awareness of minority languages, which helps protect cultural heritage and increase visibility of marginalized communities. Making this mapping free to the public could also lead to inclusivity in linguistic academic conversations and possibly even policy decisions. Finally, highlighting the world's linguistic diversity can help bolster attempts at increased equality.

4.3 Sustainability and Maintenance

Sustainability and ongoing maintenance are crucial to ensure the continued relevance and usability of the language map as a tool for global language exploration. However, since this project is a bachelor's thesis with limited resources, certain strategies and considerations for maintaining and improving the map may need to be adjusted due to time and budget constraints. Although the following sections outline the ideal steps for long-term maintenance, some of these may not be feasible within the scope of this project.

4.3.1 Data Updates

As the external databases Wikidata and OpenStreetMap are regularly updated, ensuring that the map's data remains current is important. Ideally, periodic synchronization with these databases would be set up to reflect changes, such as the addition of new languages or changes in the geographical distribution of existing languages. However, given the time and resource limitations of this project,

implementing an automated update process would require additional development and infrastructure, which is not feasible at this stage. As a result, manual updates may be necessary unless other resources become available in the future.

4.3.2 Database Management

Given that the performance of the map depends on the efficient handling of large datasets, it is important to have a system capable of managing a growing volume of data. While the number of individual languages may not increase drastically, the overall dataset can expand significantly in other ways, for example by including more detailed metadata, adding linguistic features such as translations or historical overlays. This would increase both the storage requirements and the complexity of the database queries.

Ideally, scalable solutions such as cloud storage or distributed databases would be used to ensure that the application can grow in size without significant performance degradation. However, due to the project's budget constraints, implementing such scalable solutions could incur additional costs and ongoing operational fees, which are beyond the scope of this thesis. As a result, more cost-effective alternatives may need to be explored, such as optimizing the current database structure for better performance without additional infrastructure investment.

4.3.3 Performance Optimization

As the number of languages and regions increases, the performance of the map may degrade, particularly about rendering detailed geographic polygons in the frontend. To maintain smooth performance, optimization strategies, including further refinement of polygon simplification techniques and potentially the use of more efficient data formats such as vector tiles or FlatGeobuf, would be ideal. However, since the project currently relies on GeoJSON as the primary data format throughout all stages, from the frontend rendering to the backend storage, implementing such optimizations would require a significant restructuring of the entire system. Switching to more performance-efficient formats like vector tiles would necessitate extensive changes to both the architecture of the frontend and the architecture of the backend, including the integration of new libraries and potentially migrating the database to support these new formats.

Given the time and resource limitations of this project, implementing these advanced optimization strategies is not feasible at this stage. As a result, performance monitoring tools may need to be simplified, and the more complex optimization strategies using formats like vector tiles or FlatGeobuf may need to be deferred for future iterations of the map.

4.3.4 User Feedback and Iterative Improvement

To maintain the relevance and usability of the map, regular user feedback should be collected and analyzed. While this project includes initial user testing, ongoing feedback collection and iterative development are critical for ensuring continuous improvement. Due to the project's time and resource constraints, additional rounds of testing to make a continuous feedback loop may be challenging to implement. Therefore, the scope of user feedback will be limited to the initial round conducted during the project.

4.4 Future Research Questions

The scope of the language metadata in the current version of the map is limited, since the focus is the number of speakers, language families, and geographic distribution. Extending the amount of metadata is a realistic and straightforward way to extend the practical usefulness of this project. The inclusion of some words translated into each language in the language map would improve the language map's capability as an educational tool. This could also be used to perform additional research into historical connections of languages, depending on the similarity of the various translations. Including endangerment levels of languages could provide a useful overview of the regions that need additional resources to preserve these languages.

Another way to build on top of this project is to create a language map containing historical languages and their time periods. This map could then be used as an educational tool and as a database providing a visual overview of its contents. A map like that could further be connected to the language map of this project in order to show connections between historical languages, their spread, and eventual effect on the languages that still exist today.

Although the resulting language map successfully displays the total number of speakers for a language, it would be informative to get information about the number of speakers in each region where the language is spoken. Providing information about the number of speakers on a region level would give the user a deeper understanding on how the language is spread across the world. However, implementing this feature proved challenging due to significant gaps in the existing database. We considered several workarounds, for example, integrating data from an alternative database, but this option was ultimately not feasible. It would have required extensive time to both locate a suitable, freely available dataset and integrate it effectively.

In the future, several steps could be taken to improve the lack of data and the overall accuracy of the language data. Partnerships with linguistic research institutions could provide access to more comprehensive sociolinguistic datasets. Additionally, a crowd-sourcing feature could let users contribute regional speaker estimates, which could then be verified to enhance the map's accuracy.

5

Conclusion

This projects aim was to develop an interactive language map that visualize the global distribution of spoken languages across the world. By integrating linguistic data from Wikidata with geographical data from OpenStreetMap, the resulting map lets the user search for languages, display where it is spoken, and access relevant metadata such as the number of speakers and official status.

The main objectives for the product were successfully met. The application provides a functional interface that allows users to interactively explore language data through a responsive map. The application allows users to search for languages and display their distribution on the world map, and access metadata such as the number of speakers and language family.

However, the accuracy and completeness of the visualized data are limited by the quality of the external sources used. This limitation was acknowledged within the project scope. Potential future improvements could include additional linguistic data, and integration with other linguistic databases could lead to covering gaps with missing data.

In conclusion, the project delivers an interactive language map that addresses the project's objective and purpose.

Bibliography

[1] A. Raines, "Words on a Map: The Cartography of Language", *Library of Congress*, Feb. 2022. [Online]. Available: https://blogs.loc.gov/maps/2022/02/words-on-a-map-the-cartography-of-language/, Accessed: 2025-01-28.

[2] S. Bahry, "Towards "Mapping" a Complex Language Ecology: The Case of Central Asia," in *Handbook of the Changing World Language Map,* 1st ed., S. D. Brunn, R. Kehrein, Ed., Cham, Switzerland: Springer, 2020, ch. 1, pp. 2–5. DOI: 10.1007/978-3-030-02438-3.

[3] A. Lameli, "Linguistic atlases - Traditional and modern," in *Volume 1 Theories and Methods: An International Handbook of Linguistic Variation*, P. Auer and J. E. Schmidt, Eds., Berlin, New York: De Gruyter Mouton, 2010, ch. 32, pp. 567–592. DOI: 10.1515/9783110220278.567.

[4] M. A. Bravante and W. N. Holden, "Austronesian Archipelagic Linguistic Diversity Amid Globalization in the Philippines," in *Handbook of the Changing World Language Map,* 1st ed., S. D. Brunn, R. Kehrein, Ed., Cham, Switzerland: Springer, 2020, ch. 2, p. 46. DOI: 10.1007/978-3-030-02438-3.

[5] Ethnologue, ″Ethnologue, Languages of the World ″, 2025. [Online]. Available: https://www.ethnologue.com/, Accessed: 2025-01-28.

[6] Glottolog, ″Glottolog - Comprehensive Reference for the World's Languages ″, 2025. [Online]. Available: https://glottolog.org, Accessed: 2025-01-29.

[7] Mutur Zikin, ″Muturzikin ″, 2007. [Online]. Available: http://www.muturzikin.com, Accessed: 2025-01-28.

[8] Wikipedia, ″Distribution of languages in the world ″, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Template:Distribution_of_languages_in_the_world, Accessed: 2025-01-29.

[9] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, pp. 78–85, Sep. 2014. DOI: 10.1145/2629489.

[10] J. Lovett, "Turning science into fiction? censoring population research in the soviet union, 1964–1982," *Contemporary European History*, vol. 33, pp. 192–211, Feb. 2022. DOI: 10.1017/S0960777322000054.

[11] M. Haklay and P. Weber, “Openstreetmap: User-generated street maps,” *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008. DOI: 10.1109/MPRV.2008.80.

[12] V. Fromkin, R. Rodman, and N. Hyams, "An Introduction to Language", 11th ed. Cengage Learning, 2018, [Online]. Available: https://ukhtt3nee.wordpress.com/wp-content/uploads/2019/04/an_introduction_to_language.pdf, Accessed on: 2025-05-19.

[13] A. Piscopo and E. Simperl, “Who models the world? collaborative ontology creation and user roles in wikidata,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, Nov. 2018. DOI: 10.1145/3274410.

[14] M. Faerber, F. Bartscherer, C. Menne, and A. Rettinger, “Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago,” *Semantic Web*, vol. 9, pp. 1–53, Mar. 2017. DOI: 10.3233/SW-170275.

[15] K. Chang, *Introduction to Geographic Information Systems [9th ed.]* McGraw-Hill Education, 2019, p. 47, ISBN: 9781260092585.

[16] Esri, "GIS Dictionary - Polygon", n.d. [Online]. Available: https://support.esri.com/en-us/gis-dictionary/polygon, Accessed: 2025-05-11.

[17] OpenStreetMap, "About OpenStreetMap", 2025. [Online]. Available: https://www.openstreetmap.org/about, Accessed: 2025-04-06.

[18] Nominatim, "Nominatim API Documentation", n.d. [Online]. Available: https://nominatim.org/release-docs/develop/api/Lookup/, Accessed: 2025-05-01.

[19] M. Goodwin, "What is an API?", *IBM*, Apr. 2024. [Online]. Available: https://www.ibm.com/think/topics/api, Accessed: 2025-05-10.

[20] LocationIQ, "OSM-ID", 2025. [Online]. Available: https://locationiq.com/glossary/osm-id, Accessed: 2025-05-01.

[21] OSMFoundation, "Nominatim Usage Policy (aka Geocoding Policy)", n.d. [Online]. Available: https://operations.osmfoundation.org/policies/nominatim/, Accessed: 2025-05-01.

[22] MDN web docs, "HTTP", May. 2025. [Online]. https://developer.mozilla.org/en-US/docs/Web/HTTP, Accessed: 2025-05-10.

[23] MDN web docs, "HTTP request methods", May. 2025. [Online]. https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods, Accessed: 2025-05-11.

[24] M. Goodwin and D. Nosowitz, "What is an API endpoint?," *IBM*, Aug. 2024. [Online]. https://www.ibm.com/think/topics/api-endpoint, Accessed: 2025-05-10.

[25] MDN web docs, "What is a URL?" Apr. 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_URL, Accessed: 2025-05-10.

[26] M. Yankulov, "What Is SPARQL?", *ontotext,* n.d. [Online]. https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/, Accessed: 2025-05-11.

[27] SIL International, "About ISO 639-3", n.d. [Online]. Available: https://iso639-3.sil.org/about, Accessed: 2025-04-19.

[28] SIL International, "ISO 639-3 Downloads", n.d. [Online]. Available: https://iso639-3.sil.org/code_tables/download_tables, Accessed: 2025-03-19.

[29] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and S. Hagen, *The geojson format*, Request for Comments, RFC 7946, Aug. 2016. DOI: 10.17487/RFC7946.

[30] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112–122, Oct. 1973. DOI: 10.3138/FM57-6770-U75U-7727.

[31] OpenStreetMap, "Precision of coordinates", 2024. [Online]. Available: https://wiki.openstreetmap.org/wiki/Precision_of_coordinates, Accessed: 2025-04-15.

[32] Wikipedia, "Category:Retired ISO 639-3 codes", Oct. 2024. [Online]. Available: https://en.wikipedia.org/wiki/Category:Retired_ISO_639-3_codes, Accessed: 2025-03-05.

[33] MongoDB, "MongoDB Documentation", 2025. [Online]. Available: https://www.mongodb.com/docs/manual/, Accessed: 2025-03-28.

[34] NestJS, "NestJS Documentation", 2025. [Online]. Available: https://docs.nestjs.com/, Accessed: 2025-03-25.

[35] OpenJS Foundation, "Node.js", 2025. [Online]. Available: https://nodejs.org/en, Accessed: 2025-05-18.

[36] NestJS, "Controllers - NestJS Documentation", 2024. [Online]. Available: https://docs.nestjs.com/controllers, Accessed: 2025-04-30.

[37] Internet Engineering Task Force (IETF), "The GeoJSON Format", 2016. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7946, Accessed: 2025-03-22.

[38] MongoDB, Inc., "GeoJSON Objects — MongoDB Manual", 2025. [Online]. Available: https://www.mongodb.com/docs/manual/reference/geojson/, Accessed: 2025-05-19.

[39] MongoDB, Inc., "MongoDB Manual: $regex", 2025. [Online]. Available: https://www.mongodb.com/docs/manual/reference/operator/query/regex/, Accessed: 2025-05-08.

[40] NestJS, "MongoDB - Mongoose", 2024. [Online]. Available: https://docs.nestjs.com/techniques/mongodb, Accessed: 2025-05-01.

[41] I. Zusko, "vite-template-react-ts-jest", *GitHub*, Oct. 2024. [Online]. Available: https://github.com/ivanzusko/vite-template-react-ts-jest, Accessed: 2025-04-22.

[42] Vite, "Why Vite", 2025. [Online]. Available: https://vite.dev/guide/why.html, Accessed: 2025-03-28.

[43] Meta Open Source, "Learn React – React Documentation", 2024. [Online]. Available: https://react.dev/learn, Accessed: 2025-03-20.

[44] Typescript, "What is Typescript?" 2025. [Online]. Available: https://www.typescriptlang.org/, Accessed: 2025-03-31.

[45] Render, "Web Services", n.d. [Online]. Available: https://render.com/docs/web-services, Accessed: 2025-05-15.

[46] MUI, "Material UI - React components for faster and easier web development", n.d. [Online]. Available: https://mui.com/material-ui/, Accessed: 2025-05-01.

[47] MapLibre Project, "MapLibre - Open-source mapping libraries for web and mobile apps", n.d. [Online]. Available: https://maplibre.org, Accessed: 2025-05-19.

[48] H. Djirdeh, "React Basics: How and When to Use React Context", *Telerik*, 2023. [Online]. Available: https://www.telerik.com/blogs/react-basics-how-when-use-react-context, Accessed: 2025-05-13.

[49] MapLibre Project, "MapLibre - GL JS API", n.d. [Online]. Available: https://maplibre.org/maplibre-gl-js/docs/API/, Accessed: 2025-05-19.

[50] M. Halim, "Understanding UI Design Patterns and Why They Matter", *webfeeling*, Apr. 2024. [Online]. https://webfeeling.es/understanding-ui-design-patterns-and-why-they-matter/, Accessed: 2025-05-02.

[51] UI-Patterns, "User Interface Design patterns", 2025. [Online]. https://ui-patterns.com, Accessed: 2025-04-08.

[52] Cloudfare, "Why is HTTP not secure? | HTTP vs. HTTPS", n.d. [Online]. Available: https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/, Accessed: 2025-05-19.

[53] C. M. Barnum, *Usability Testing Essentials: Ready, Set...Test!* Morgan Kaufmann, 2011, ISBN: 9780123750921.

[54] MongoDB, Inc., "JSON and BSON", 2025. [Online]. Available: https://www.mongodb.com/resources/basics/json-and-bson, Accessed: 2025-05-19.

[55] Esri, "What is KML?—ArcGIS Pro Documentation", 2024. [Online]. Available: https://pro.arcgis.com/en/pro-app/latest/help/data/kml/what-is-kml-.htm, Accessed: 2025-03-20.

[56] M. H. Lilleengen, "FlatGeobuf", 2021. [Online]. Available: https://flatgeobuf.org/, Accessed: 2025-05-02.

[57] Mapbox, "Vector tiles introduction", 2025. [Online]. Available: https://docs.mapbox.com/data/tilesets/guides/vector-tiles-introduction/, Accessed: 2025-05-02.

[58] D. Sharma, *Language and Borders*. Oxford University Press, Jul. 2023. DOI: 10.1093/acrefore/9780190846626.013.722.

A

Appendix 1

A.1 SPARQL Queries for Language Data

Listing A.1: SPARQL Query for Langauges and Their Metadata

```
SELECT ?language ?languageLabel ?iso639_3 ?instanceOfLabel
    ?immediate_language_family_Label ?number_of_speakers
    ?nos_place_Label ?nos_applies_to_Label ?nos_time

WHERE {
  ?language wdt:P220 ?iso639_3 .
  ?language wdt:P31 ?instanceOf .

  OPTIONAL {
    ?language p:P1098 ?number_of_speakers_statement.
    ?number_of_speakers_statement ps:P1098 ?number_of_speakers.
    OPTIONAL { ?number_of_speakers_statement pq:P3005 ?nos_place }.
    OPTIONAL { ?number_of_speakers_statement pq:P518 ?nos_applies_to }.
    OPTIONAL { ?number_of_speakers_statement pq:P585 ?nos_time }.
  }.

  OPTIONAL {
    ?language wdt:P279 ?immediate_language_family_.
    ?immediate_language_family_ wdt:P31 wd:Q25295
  }.

  SERVICE wikibase:label { bd:serviceParam wikibase:language
    "en, [AUTO_LANGUAGE]" }
}
ORDER BY ASC(UCASE(str(?iso639_3)))
```

Listing A.2: SPARQL Query for Regional Data

```
SELECT ?language ?languageLabel ?iso_code ?region ?regionLabel
    ?country ?countryLabel ?region_osm_id ?country_osm_id
    ?isOfficial ?country_of_regionLabel

WHERE {{
    VALUES ?iso_code {iso_array}  # Feeds in 200 at a time

    ?language wdt:P220 ?iso_code.
    OPTIONAL {{ ?language wdt:P17 ?country.
        OPTIONAL {{ ?country wdt:P37 ?official_lang. }}
        OPTIONAL {{ ?country wdt:P402 ?country_osm_id }}
    }}
    BIND( ?language = ?official_lang AS ?isOfficial )
    OPTIONAL {{ ?language wdt:P2341 ?region.
        OPTIONAL {{ ?region wdt:P17 ?country_of_region. }}
        OPTIONAL {{ ?region wdt:P402 ?region_osm_id. }}
    }}

    SERVICE wikibase:label {{ bd:serviceParam wikibase:language
        "en,[AUTO_LANGUAGE]". }}
}}
```