CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Swimming Through Data

An Open Source Tool for Creating Interactive Heatmaps of Geo-Spatial Data

Bachelor's thesis in Computer Science and Engineering

Benjamin Billton
Leona Geelnard
Rasmus Helgesson
Oskar Sköld
Simon Uggla
Hugo Unosson

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

Bachelor's thesis 2025

Swimming Through Data

An Open Source Tool for Creating Interactive Heatmaps of Geo-Spatial Data

Benjamin Billton
Leona Geelnard
Rasmus Helgesson
Oskar Sköld
Simon Uggla
Hugo Unosson

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

Bachelor's thesis template for LaTeX for CSE
An Open Source Tool for Creating Interactive Heatmaps of Geo-Spatial Data
Benjamin Billton Leona Geelnard Rasmus Helgesson Oskar Sköld Simon Uggla
Hugo Unosson

Supervisor (Handledare): Jasmina Maric, Interaction Design and Software Engineering
Examiners: Arne Linde and Patrik Jansson, Department of Computer Science and Engineering
Graded by teacher (Rättande lärare): Thommy Eriksson, Interaction Design and Software 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

Typeset in LaTeX
Gothenburg, Sweden 2025

Abstract

Tracking aquatic animal movements, such as those of the Atlantic sturgeon (*Acipenser oxyrinchus*), presents significant challenges due to sparse data from acoustic sensors and a lack of accessible visualization methods. This project addresses these issues by developing "Swimming Through Data", an open-source, interactive tool for visualizing geo-spatial tracking data. The primary aim was to create a system that generates clear, accurate heatmaps for both popular science audiences and scientific researchers, without misrepresenting the underlying data. Key objectives included refining raw sensor data, building a durable and extensible software architecture, enabling pedagogical data representation, and ensuring the tool's public availability and reusability.

The methodology involved developing a backend architecture with a RESTful API and a data management pipeline for processing sensor data. The frontend was built using React, with Leaflet and Leaflet.heat for interactive map displays. A user-centered design process, incorporating the Triple Diamond approach and MoSCoW analysis, guided development. The tool's effectiveness and usability were evaluated through qualitative user tests and heuristic evaluations, focusing on learnability, efficiency, memorability, error handling, and user satisfaction.

The project successfully produced a functional tool capable of data uploading, processing, and interactive heatmap visualization. User testing revealed high overall satisfaction and good learnability, alongside feedback for UI enhancements. The visualization approach was found to be clear and effective in representing sparse data.

Sammandrag

Att spåra vattenlevande djurs rörelser, såsom de hos den atlantiska stören (*Acipenser oxyrinchus*), innebär utmaningar på grund av temporalt och spatialt lågupplöst data från akustiska sensorer och brist på tillgängliga visualiseringsmetoder. Detta projekt hanterar dessa problem genom att utveckla "Swimming Through Data", ett interaktivt verktyg med öppen källkod för att visualisera geo-spatial spårningsdata. Det primära syftet var att skapa ett system som genererar tydliga, exakta värmekartor för både populärvetenskaplig publik och vetenskapliga forskare, utan att förvränga de underliggande data. Huvudmålen var att förfina sensordata, bygga en hållbar och utbyggbar mjukvaruarkitektur, möjliggöra pedagogisk datarepresentation och att säkerställa verktygets offentliga tillgänglighet och återanvändbarhet.

Metodiken innebar att utveckla en backend-arkitektur med ett RESTful API och en datahanterings-pipeline för bearbetning av sensordata. Frontenden byggdes med hjälp av React, Leaflet och Leaflet.heat för interaktiva värmekartor. En användarcentrerad designprocess, som inkluderar Triple Diamond-metoden och MoSCoW-analys, vägledde utvecklingen. Verktygets effektivitet och användbarhet utvärderades genom kvalitativa användartester och heuristiska utvärderingar, med fokus på lärbarhet, effektivitet, minnesbarhet, felhantering och användarnöjdhet.

Projektet producerade framgångsrikt ett funktionellt verktyg som kan ladda upp data, bearbeta och interaktiv värmekartvisualisering. Användartestning visade hög övergripande tillfredsställelse och god inlärningsbarhet, tillsammans med feedback för förbättringar av användargränssnittet. Visualiseringsmetoden visade sig vara tydlig och effektiv för att representera glesa data.

Acknowledgements

Dan Calderon - thanks for generously sharing your insights and expertise regarding the reintroduction of the Atlantic sturgeon in Göta älv. Your willingness to discuss 'The Return of the Sturgeon' project provided invaluable context and inspiration for our thesis. **Johan Leander** - thanks for the help and access to the data. **Johan Höjesjö** - thanks for helping us connect with similar ongoing projects. **Susan Gotensparre** - thanks for the help in finding the right people and helping us to give the project a more popular scientific perspective.

Benjamin Billton, Leona Geelnard, Rasmus Helgesson, Oskar Sköld, Simon Uggla, Hugo Unosson, Gothenburg, June 2025

Contents

List of Figures

List of Tables

1

Introduction

There are many projects in which scientists face the issue of tracking animal movements without having continuous data about it; more specifically, this is often the case in fish research. Examples of ecological insights they study include seasonal migrations, feeding areas, spawning sites, and stock assessments of managed species [12]. In the case of fish tracking, the water blocks GPS signals, and in most cases, acoustic sensors are used instead [40]. These sensors have limited reach, leaving gaps in the data and limiting how much you can say about the behavior of these animals. In addition, it can be difficult to present the findings in a comprehensive way.

This project was created upon identifying this problem in a specific fish research project, which is ongoing in the river Göta älv, Sweden, which is focused on reintroducing the Atlantic sturgeon into these waters. However, there are various similar projects in other countries (such as Germany [14], Poland [18], and countries around the Baltic area [5]) that could face similar problems. If scientists want to participate in international collaborations, it would be beneficial to have a standardized way to store and visualize the data.

The Atlantic sturgeon (Atlantstör) is a fish that has not existed in Sweden since the late 1800s and is now classified as Regionally Extinct. "The Return of the Sturgeon" (Störens återkomst) is the above-mentioned research project, operated and managed by Sweden's Sportfishing and Fishery Conservation Association (Sveriges Sportfiske- och Fiskevårdsförbund) to reestablish the gracious Atlantic sturgeon within Göta älv and Nordre älv. During the first half of 2024, a total of 95 tagged sturgeons have been released [25]. Using approximately 25 tactically placed underwater sensors that log whenever a given tag has been sensed, some rough movement patterns can be gathered. However, this data is quite difficult to understand, and an easy-to-use data visualization program is a highly prioritized demand.

This bachelor's project is intended to facilitate future research and knowledge within fish tracking. The project would help this purpose through a) making it easier to communicate progress and purpose to potential sponsors, which plausibly can result in more financial support for such projects, b) making it easier to spread the knowledge within our society through popular science events by having well-visualized data, and maybe c) making it easier for researchers themselves to use the data without spending much time on dealing with the raw data itself.

On a higher level, the increased knowledge and facilitated research within fish tracking can have various positive impacts such as increasing opportunities for re-introducing extinct species or monitoring red-listed species to protect them, which works toward values of biodiversity and moral values of taking responsibility for the harm humankind has caused said species through our way of life (pollution, destruction of habitats, over-fishing, etc.) [4].

Main purpose

We aim to create a visualization tool for fish tracking data that generates visualizations that are as clear as possible for popular science audiences without showing inaccurate or misleading data (i.e. not making strong assumptions), and that can also work as an analysis tool for scientists.

Sub purposes

In order to achieve a visualization tool that lives up to the above-stated aim, the following steps should be explored.

- Refine the current form of raw data into something more accessible.
- Make the software infrastructure durable over time and easy to build upon for others in the future.
- The tool should enable pedagogical presentations of scientific data on fish tracking
- Make the tool publicly available and reusable in different projects.

1.1 Limitations

Due to the time constraints and the program's specific requirements, it is decided not to prioritize advanced security measures or a highly customizable UI. The primary focus is on enabling the functionality to upload data, but not necessarily on creating the optimal layout for this feature. Instead, the focus is on developing the data analysis tool and the data structuring necessary for future technological advancements.

The application is designed to be user-friendly for non-technical individuals, enabling them to easily integrate the application into their workflows. Since the project will ultimately be open-source, it allows others to contribute by enhancing or creating newer versions. Consequently, certain features, such as user registration, session management, authentication, or authorization, are not implemented in this iteration. The admin page, where you control the analysis and uploaded data, is instead a route on the frontend that uses unsecured and unauthenticated API calls, simplifying the initial implementation. This approach leaves the application vulnerable to attacks. It will be crucial to secure these API endpoints and admin routes if this application is publicly accessible.

1.1.1 Data Management Restrictions

Current client applications on the world wide web utilize some sort of menu for different options in terms of data processing. Our goal in the data management application is to create software that can recognize different types of data by itself and process the information dependent on the provided data. This results in an improvement in terms of user experience and minimizes unnecessary components in the user interface.

Because of technological and programmable restrictions, the data management application is semi-general and is optimized for this specific use case, but it will be possible to improve it through future development.

The problem is caused by how regex technology and artificial intelligence currently function and the different requirements that need to be fulfilled in development (e.g. expensive hardware, larger amounts of provided data, hard-coded sequences of text, and the limitations of regex patterns).

2

Theory and Background

This section will describe important words and give context about concepts used in this project. Starting with the workflow used, followed by information about the fish and its habitat, which is needed to interpret the data. After this, a literature study is presented, followed by explanations of the technology used in the backend and frontend of the program. Finally, an analysis and test method are explained.

2.1 The Triple Diamond Approach

The task to create an easy-to-understand visualization of the data is not trivial as there are no objectively correct solutions to this problem. Therefore, a dynamic problem-solving framework is beneficial to produce relevant results in the given time frame [3]. This is the reason that the Triple Diamond approach is useful in this case.

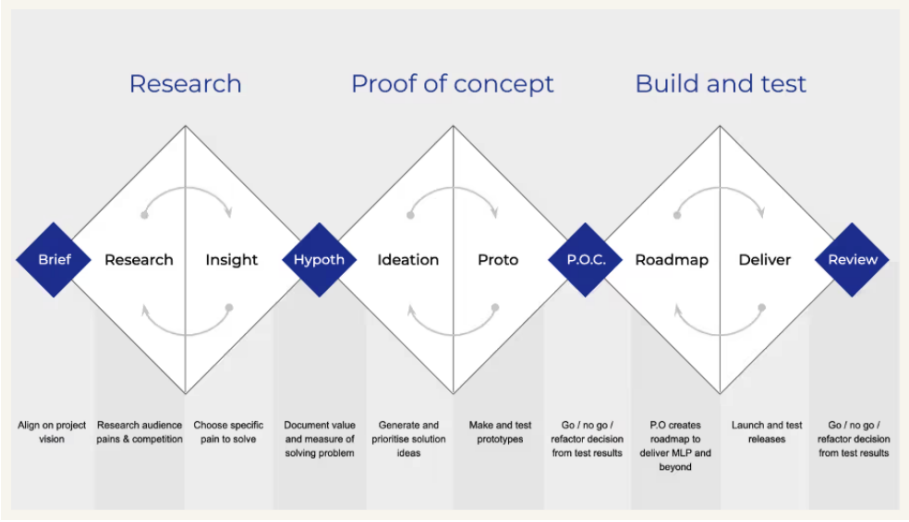


Figure 2.1: Graphical summary of the Triple Diamond approach [3]

In figure 2.1 a graphical overview of the Triple Diamond Approach can be viewed. The process consists of three main stages: research; proof of concept; and build and test.

2.2 The Atlantic Sturgeon

The Atlantic sturgeon (*Acipenser oxyrhinchus*) is a species of fish that has existed in northwestern Europe, alongside the European sturgeon [2]. Archaeologic findings show that it has existed in the Baltic Sea for at least 5000 years, and existing evidence points to the fact that the Atlantic sturgeon was the sturgeon species that historically dominated Swedish waters. It is also confirmed that this species has used Göta älv as a place for mating previously, as there are small sturgeons from this species in the collections of Gothenburg Natural History Museum from the 19th century too young to have left the area where they were born. This is significant as this species returns to the area in which it was born later in life to mate, around every year after reaching a mature age. However, due to human impacts and exploitation, the Atlantic sturgeon was deemed functionally extinct in Swedish waters in 1970, and the last known capture of an Atlantic sturgeon in Swedish waters before that was in 1948.

The Atlantic sturgeon can, according to historical evidence, live to an age of between 120 and 140 years (which, however, is rare today because of exploitation) [2]. In its lifetime, it can grow to a length of over 5 meters and a weight of over 600 kg. They live their life in both freshwater and saltwater, being born in freshwater and spending most of their adult life in saltwater, apart from when they are mature and migrate back to the freshwater areas around every year to mate [41]. They are a bottom-feeding species, spending a lot of their time searching for food at the bottom.

2.3 Göta älv

Göta älv is 93 kilometers long from its start in Vänern to the sea in Gothenburg, with a typical breadth of 50 to 400 meters [42]. The depth is above 6.3 meters, and can in some areas be up to 20 meters. The river divides into two branches around Hisingen. The total outflow from Göta älv to the sea is on average 598 cubic meters per second, and the yearly (average) maximum is 1041 cubic meters per second [35]. Of this, the Göta älv branch normally stands for around $\frac{1}{3}$ of the water while Nordre älv stands for around $\frac{2}{3}$ of the water.

The northern branch, Nordre älv, is a salt wedge estuary. This means that it has the majority of its freshwater outflow in the upper layer, while the seawater flows in along the floor of the river and is prevented from mixing with the freshwater. It also means that it has a low tidal influence and a large breadth-to-depth ratio. In between the top and bottom layers, there is a halocline with more mixed water, and this means that freshwater species as well as brackish and saltwater species inhabit the estuary.

2.4 The sensor data

The sensors' data is structured and consists of certain parameters describing time, the animal's behavior, the speed of the animal at a specific time, latitude and longitude, and the animal's tag. The sensor has a radius that can vary depending on the current environmental conditions, such as wind, position, water speed, and water clarity. This creates an uncertainty in the data that needs to be taken care of.

The data is extracted from the sensor in a manual way, where the extractor physically accesses the specific sensor and saves the data into the computer. It then gets processed by SLU into structured CSV and Excel files for later use. The data itself is not filtered or transformed in any way. This results in unnecessary information in the provided data.

2.5 Previous Research

A literature study about previous research was conducted to obtain an overview of current research regarding modeling and visualization of spatial data. As the data of this project is geo-spatial, changes over time, and has a low temporal resolution, the literature study focuses on projects with data that fulfill multiple of these attributes. The following three solutions are compared in a paper released in the journal Animal Biotelemetry and seem to be common solutions to this kind of problem [12].

One approach to estimate the fish position is *raw data methods*. This means that the fish are assumed to stay at the positions of the last detections until a new detection is logged. Here the visualizations often consist of scatter plots, as well as graphs of coordinates as a function of time.

Another method is to use *Center of Activity (COA)* where a linear or non-linear interpolation is performed on the spatial data to estimate the fish position in between detections. Moreover, a weighted average using the amount of detections for a sensor enables a more accurate estimation. The visualizations made from this method can be maps with lines and dots that move with time.

A third method is *Dynamic Brownian bridge movement models (dBBMM)*. To estimate the position in between detections, multiple Brownian bridges are simulated. A Brownian bridge is a random walk between two points where the start and end positions are known. By simulating multiple of these, a probability density distribution is created [29]. In this case, heatmaps are often created to visualize the probability density function using different colors to represent intensity.

To compare the models, they simulated fish movement representative of real data and recorded detections. Using only the detections, they performed estimations of the fish position using the different models. An efficiency score was created to quantify how close to the "real" data the models were. The result was that COA-models were by far the most accurate, with dBBMM-models in second place. The

raw data methods were not nearly as accurate as the other two.

2.6 APIs

The backend architecture that connects the data management pipeline, database, calculation tools, and the user interface uses a RESTful API. A detailed description can be found in Appendix C.

2.7 Data Management Pipeline

Data management pipelines are used to structure, filter, and convert different data types. Think of an automated assembly line for data. The assembly line takes raw data from a source, prepares it, and then loads it into the SQL database in the correct format and structure [32].

2.7.1 Different potential stages in the data pipeline:

Source: A source is typically raw data that is unfiltered and difficult to use for larger applications. It could be anything from simple files (e.g. CSV, Excel, JSON files), data streams from applications, data from other databases, or data pulled from APIs. The pipeline's role is to know where to find the data and how to access it [32].

Extraction: This is the stage where the data is being read into the program. The pipeline executes necessary commands or operations to grab the data. It is like picking up the raw materials from the loading dock in our assembly analogy [32].

Transformation: This stage is the most complex. Raw data is rarely in the perfect state to go directly into your structured SQL tables. Transformation involves cleaning, shaping, and preparing the data [32].

Cleaning: Involves handling missing values (e.g. replacing or removing non-defined values such as blanks, N/A, NaN, or None values dependent on the programming language). It also involves standardizing values for standard use cases in the internal resources across the API [32].

Formatting: Here is where we convert the data to standard types that the program can handle and so that the data can be injected into the SQL database [32].

Structuring: Structuring includes renaming columns to match your SQL table schema, combining data from multiple source columns into one, and splitting one source column into multiple SQL columns [32].

Validating: The stage of validating is usually for type checking, this makes sure that the right variable types are being inputted to the SQL database [32].

Why do we use a Pipeline?
A data pipeline creates automation; it can run on a schedule without manual intervention. This creates reliability because it is a repeatable process which in turn creates consistency where the data goes through the same cleaning and transformation steps every time. This results in scalability where the pipeline can be designed to handle growing volumes of data efficiently. The structure of pipelines allows for easier updating and troubleshooting of the data processing, something that is useful in a program that otherwise could become complex and difficult to manage.

2.8 User Interface

Choosing an appropriate tech stack for the frontend is crucial due to the impact it has on the overall performance of the final application.[17] This project focuses on visualizing data; with that in mind, the following tech stack has been considered.

2.8.1 React

React is a JavaScript library that is used to create the graphical user interface for web applications [13]. It uses a component-based code structure, meaning that one large component, such as a tab on the website, can be broken down into many smaller components. Components can be reused and are only updated when needed. For instance, a button could be coded as one small component that, on click, updates another component. React handles this efficiently; it will create a virtual user interface before and after clicking the button. React will then notice the differences between the user interfaces and update only the components that have been changed instead of the whole tab. This makes visual updates on the website fast, which is much needed in this project's web application.

2.8.2 Leaflet

Leaflet is a JavaScript library [20]. It is used to create an interactive map with standard features such as pan and zoom. Leaflet is easy to style to fit the needs of whoever is using it. Leaflet has a lot of plugins that can be added if needed. This project uses one of these plugins. It is called "Leaflet.heat" and is used to add a heatmap layer to the map [21].

2.8.3 Heatmap

A heatmap is a technique to visualize 2-dimensional data [6]. It uses a matrix filled with numeric values, typically a range of zero to one, and converts each cell to a color. The color of each cell can then be blended or blurred to create a high-resolution image of a cloud of colors. This image can be used as an overlay on a map to visualize positional data.

A typical example of heatmap visualization would be to present the amount of rain that will fall from a raincloud. Such a visualization can be found at SMHI [34].

A blue to red gradient is used to represent the amount of expected rain. In this project's case, the heatmap is used to visualize the expected probability of a sturgeon being at a certain position at a certain time.

2.8.4 Accessibility

Proper accessibility features are some of the key features in achieving an intuitive visualization tool. Some examples of such features include keyboard navigation, high contrast between colors, or tooltips. What features to implement is up to the developer to decide; it is important to choose features that bring value to the user. In the development of this project, three accessibility features have been in focus:

Tooltips
A tooltip is a small box with text that appears while hovering over an element. The text should describe the purpose of the element or a certain action that is connected to that element.

User feedback
Giving the user feedback after certain events have happened is a way to inform the user whether the event was successful or not. For instance, text messages can be displayed with a red background to symbolize that an error occurred, while a green background symbolizes success.

Choice of color
When visualizing data with color, such as a heatmap, it is important to keep in mind that everyone perceives color differently. Therefore, it can be a good option to offer a selection of different color schemes. Especially, a heatmap that relies on visualization via colors needs to have colorblind-friendly options.

These accessibility features will hopefully make it easier to use and consume the website for a wide range of users.

2.9 MoSCoW analysis

MoSCoW analysis is a method that can be used for prioritizing features of a product. The capitalized letters represent the key words of the analysis.[19] The M stands for Must Have, which categorizes the features that the product absolutely must have in order to be valuable. The S stands for Should Have, which indicates a feature that would be very beneficial to have and should be implemented if it is possible, but it is not a Must Have because the product can still function and have value without it. The C stands for Could Have, indicating a feature that would be useful, but not central to the product, and could be implemented if, for example, it is not too costly or demanding. Finally, the W stands for Won't Have/Would Have, a category of features that for some reason will not be in this version of the product, perhaps because of some constraint, but which would be good to have in the future. Features from this category are not currently needed but could migrate to the Must Have category in some future edition of the product.

2.10 User Test Methods

To test and guide design decisions, different user tests can be performed which give information about different aspects of the user experience. Listed down below are three common tests: qualitative, A/B, and heuristic tests.

2.10.1 Qualitative test

A qualitative test is beneficial to get more specific information about users' experience. By performing the test in person and having it moderated, specific questions can be raised, but to get a more clear basis that can be quantitatively compared, different categories with scores should be created [26]. There are five aspects of user experience that should be tested, which are: learnability, efficiency, memorability, errors, and satisfaction. After a moderated qualitative user test, a score between one and five is given to each aspect. The user is not the most accurate judge to give a score for all categories. Instead, the moderator can ask relevant questions and get a feeling about how well the user performed in regard to the different categories.

Learnability measures how easily the test user can navigate and use the program. Measures to increase learnability include pop-up help boxes and good documentation which allows the user to learn the program without external help. If the layout and structure of the program are self-explanatory and the user easily finds the right content, the learnability score gets high. The opposite is true for low learnability scores.

Efficiency is a score that explains how fast the user can do different tasks within the program. Even though the learnability score is high, there could exist complicated repetitions and unnecessary steps that they need to go through every time certain tasks are completed. A high efficiency score for a program means that the user can quickly perform the necessary task. One way to measure this score is to take time on main tasks and compare the time duration of different operations. This means that the test finds out which tasks stick out in completion time and therefore have the most potential for optimization. By combining the measured times with qualitative questions about how the user experienced time on different tasks, the moderator can get a feeling for what a short time is and use this as a base when setting an efficiency score.

Memorability means how eager the user is to return to the program after a session and also how easy it is to use it the next times. To set a score of memorability, the moderator can ask relevant questions to find out how the user feels about returning to the program. By measuring the time it takes to complete different tasks, in the same way as the efficiency score, the test can find out how much quicker the user is at using the program than the first time. If the user feels the need to use the program again and the program is more efficient to use the next times, there is a high memorability score.

Errors is a score that measures how well the program compensates and adjusts

for user flaws. For example, if the user exits the program during data upload or clicks buttons too quickly, the program should be able to handle potential errors. To test this, the user can be given the task to try to crash the program or do operations hastily. By having a moderator watch the actions and also by having screen recordings, they can find out which errors the program has.

Satisfaction measures how the user is feeling after exiting the program. Questions to answer could include: Could they do the things they wanted with the program, and was it a nice experience using it? This aspect functions more as an overarching score that is correlated with the other four measured aspects. It can be good to have this more general score to compare the others to. As satisfaction is very subjective, this score is mainly set by the user in assistance with the moderator.

2.10.2 A/B testing

A/B testing is often automated in a digital environment and performed on large numbers of users (in the thousands). It works by creating two versions of the same graphical interface and defining a goal that should be optimized towards [7]. For example, create two different colored buttons and see which one gets more clicks. Compared to the experience-based heuristic test, an A/B test assumes that you do not know what is a good design, but you find out by creating many versions that compete, and the final one is assumed to be better than the other designs in regard to a certain goal.

2.10.3 Heuristic test

A heuristic test is similar to a qualitative user test, but instead of testing on users, 3-5 experts in interaction are consulted [16]. This can be a great tool for detecting any major problems in the interaction design early on in the design process. The persons that are consulted in the heuristic test should have good knowledge both of communication/interaction design and the specific thing that you are trying to communicate. This ensures getting quick and science-based insights in what could be some potential usability issues of the application. However, this is not a substitute for actual user tests with the target audience, as the results from the heuristic test can be biased based on what the current paradigm is in the field, and always need to be confirmed empirically even if they have a high likelihood of being correct (it is a very good hypothesis based on research, but not a final result).

3

Method

The process of this project involved several steps, all related to the user experience. To complete the goal of creating a clear interactive visualization, there was first a decision process for the program as a whole, then there was simultaneous work in the backend (data loading/processing and analysis) and the frontend (visualization and user interface), and user tests were performed. While what would be directly seen by the user was the UI, all processes in the backend also contributed to the user experience, for example, by how fast operations were, and how well the analysis part on which the final visualization was based worked. For the user tests, the initial plan was to use a combination of qualitative and quantitative (A/B) user tests. Due to time constraints, only one of them could be completed, and it was decided to focus on the qualitative tests because they could provide us with more specific feedback about why the users enjoyed or didn't enjoy the program.

The final product was composed of different components. An overview is shown in Figure 3.1.

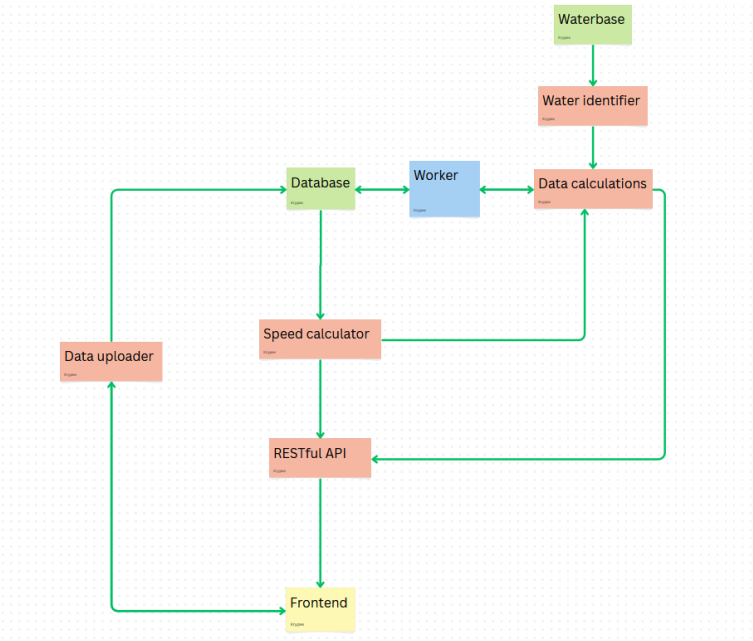


Figure 3.1: A diagram of the different components in the project

3.1 Workflow and Decision Process

Following the Triple Diamond Approach, thorough research was conducted to get a deeper understanding of the data and the problem. This included the following:

- Brainstorming within the project group and researching online. Initially, this was mainly focused on what types of visualization of the data were feasible and seemed promising for the purpose of the project.
- Consulting experts in relevant fields and people working on the project Return of the Sturgeon. See Appendix A for a summary of the people that were consulted. At this stage valuable inputs were added to the initial ideas.
- Decisions on the software infrastructure. It was decided to use a structure with a database, which stores all the relevant information and communicates with other backend components in order to perform necessary calculations. This is then sent to the frontend which shows the visualization. The components were connected as shown in 3.1.
- MoSCoW analysis. This was performed based on the information that was gathered from the aforementioned experts.
- Literature study. To get an understanding of current research in the field a literature study was conducted. This part of the research should give an understanding about how different researchers have tackled the same or similar problems to help guide the solution process and find a relevant niche focus.

3.2 MoSCoW Analysis

A MoSCoW Analysis was conducted to better understand the demands and requirements for the project. The full analysis is documented within Appendix B.

3.3 Data Uploading and Processing

The data loading component in the backend was structured as a hybrid-modular component. The component has been connected to different components, independently of how the components are programmed, while being hardwired to other components which will create a larger singular component [10].

The data uploading component was a Pipeline of operations that was executed in parallel, depending on what type of data that was provided.

Conversion and Transformation
The first operation was the conversion and transformation between different formats. The operation took a list of different data files with different data types and figured out what type of file formats they were and then executed different parts of code

to convert the file types into a standard CSV file format while also merging the provided files into a singular data structure. [32].

Classification
The second stage of the pipeline was the data classification process. This process was made to justify column names and convert them to standard column names used over the entire application. The classification method was dependent on the provided features and standard patterns or the default application features. These default parameters pattern matched different words using regex patterns, which meant that a single standard column could equal a larger amount of different words. This mechanism enabled the program to understand different types of data, where the user did not have to specify information about the data, which aimed to increase the friendliness of the application to the end-user [32].

Extraction
The third stage extracted the data based on the provided column names. This caused a decrease of data stored in the SQL database, which contributed to a cleaner database structure and data structure and also to making future operations in the migrations, application updates, or future changes in the database. This was done to improve the long-term user experience because of how the program operated [32].

Second Transformation
The fourth stage was where the new data classification was structured into a standard data structure for uploading to the database. Here the data was received from all the different pipelines to then add these into a dictionary. There were some specific data points needed to be calculated to group all the necessary information to then be able to create timestamps to display the heatmap [32].

Insertion
The fifth and last stage was to send the structured data to the database, where the data was type-checked and structured into DataInsert-Objects, which made the application able to inject the data into the database by using hard-coded SQL queries over a list of objects. The parallel execution of these pipelines, along with parallel operations within them, maximized hardware utilization, efficiency, and the speed of the uploading procedure. This approach was implemented to enhance user experience and overall efficiency [32].

3.4 Database and Database Access

A relational database was chosen for storing and managing the collected sensor data as well as application-specific information. The design of the database focused on achieving a high degree of normalization. The primary purpose of normalization was to ensure data integrity, minimize data redundancy, and consequently reduce the likelihood of data anomalies (such as insertion, update, and deletion anomalies) [23]. By structuring data to avoid unnecessary repetition and by establishing clear relationships between data entities, we aimed to create a robust and flexible database model capable of handling future changes and expansions in data volumes and types

while maintaining performance, consistency, and minimizing data-related errors.

To interact with the database, a number of Python-based helper classes were developed. These classes function as an abstraction layer between the application logic and direct database operations. By using these helper classes, SQL queries and database management logic were encapsulated, which prevented the proliferation of raw SQL calls in other parts of the code. This approach was chosen to increase the code's maintainability, readability, and to simplify the development process. Developers could interact with the database via a well-defined and consistent interface without needing to handle the complexity of SQL syntax for each individual operation, which also serves as a control point to enforce data validation rules, further contributing to the minimization of anomalies. This centralized management of database interactions also facilitates troubleshooting and future optimizations.

3.5 Analysis

The analysis component was the part that performed the calculations necessary to generate a heatmap of a specific sturgeon at a given time interval. The decision to make the component as modular as possible was made early in the development process. We chose to structure the whole analysis as a pipelined layered process, and there were several advantages to using this structure compared to other methods.

3.5.1 Frame from Layers

For one frame, i.e. the positional heatmap for one sturgeon at a specific date and time, multiple unique heatmap layers were used. Each layer had its own heatmap, and the resulting frame was the product of multiplying all layers together for that frame. Given that F^t is the final frame for time t and that L_i^t is the heatmap layer for layer i at time t, we can state the following equation for the final frame:

$$F^t = \prod_{i=1}^{n} L_i^t = L_1^t \cdot L_2^t \cdot \ldots \cdot L_n^t$$

Figure 3.2: A formula describing the final layer

Where n is the number of layers in that frame. There were two main advantages to calculating the final frame as a product. Since multiplication is commutative, the order of the frames did not matter. Moreover, if any factor in the product were zero, the entire product became zero. These advantages were used to optimize the program since there were many unnecessary calculations.

3.5.2 The Analysis Layer

The main idea was that each layer had its own assumption, defining a rule of whether a tracked object could be there or not. By combining enough of these assumptions,

a heatmap displaying approximately where the tracked object is located was aimed to be created. One of the simplest assumptions was that a tracked object needed to be in water, since we were tracking fish. This means that L_1^t was just a heatmap where only the points located in water had non-zero values.

3.6 Example Configuration of the Analysis Pipeline for a Frame

Since the aim was to build an analysis platform and not analyze sturgeons directly, we created five example layers to better explain the capabilities of the component. These layers were entirely demonstrative, exhibiting a realistic although hypothetical scenario specific to sturgeons. This implementation aimed to make it possible for users with significantly more expertise within the subject to produce their own assumptions and use them in the pipeline.

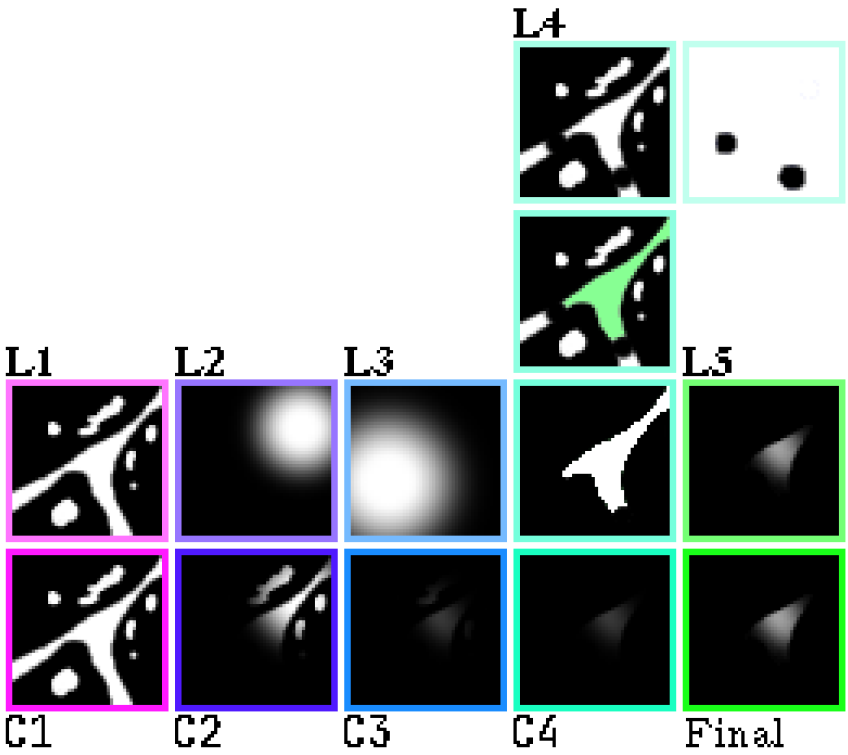


Figure 3.3: A diagram of the different layers used in the example frame generation

The figure at 3.3 depicts five example layers, L_1 to L_5 and C_1 to C_4. C_n can be described as $C_n = L_1 \cdot L_2 \cdot \ldots \cdot L_n$, where the final frame F is the same as C_5. Each frame only has values between zero and one.

3.6.1 Water identification

The first layer at 3.3, L_1 identified where there was water. The result was that any points located on land were set to zero. The assumption for this layer was that **No sturgeon can move on land**.

3.6.2 Sensor identification

L_2 and L_3 were generated using data from the sensors. Each sensor had a list of timestamps, $T = \{t_0, t_1, \ldots, t_n\}$ and their radius that they could detect within, r_{detect}. A maximum radius, r_{max}, was calculated using these values together with the current frame time, t_0, and our approximate maximum velocity v_{max}.
The first step was to find the value in T that is closest to t_0 and call this t. Keep in mind that $t_0 - t$ can be both positive and negative. This value was stored as $\Delta t = t_0 - t$. After obtaining the time difference, the maximum radius was calculated as follows:

$$r_{\text{max}} = |\Delta t| \cdot v_{\text{max}} + r_{\text{detect}}$$

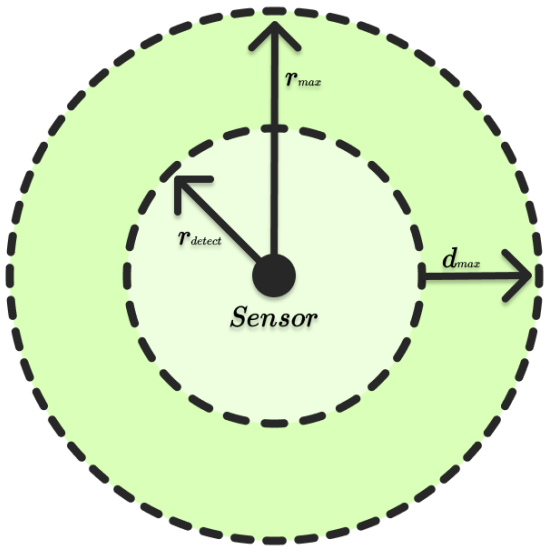


Figure 3.4: A diagram of a sensor displaying detection radius, maximum radius and maximum distance

This means that if the nearest detection was made 10 seconds ago, the maximum distance the object can move in that period is $d_{\text{max}} = 10 \cdot v_{\text{max}}$. To generate the heatmap, the distance from the sensor to each point in the heatmap was calculated and their values were set according to the distance compared to r_{max}. The assumption for the sensor layers can be expressed as **given a max velocity and**

a fixed sensor detection radius, we can guarantee that that the max distance the sturgeon will move is the max velocity multiplied with the time difference.

3.6.3 Air Locking Sensors

The fourth layer was created by assuming that **a sturgeon moving within a sensor's detection radius will be logged**. This means that for a frame no sturgeon will be able to be within all sensors except the sensor with the nearest ping, since they would in that case log a detection and therefore already be marked as having the nearest ping. A simple path-finding algorithm was then used to mark the area around the nearest sensor; this eliminated all nearby waters that were not directly accessible.

3.6.4 Flicker elimination

The final layer was a simple scalar that was applied to all the points in C_4, to reduce flickering between frames. The scalar was calculated by finding the max value of C_4 and multiplying all values with the inverse of C_4, this guaranteed that $max(F_n) = 1$.

3.7 Optimization by ordering layers

The order of the layers within a frame matters! Since calculations for coordinates that were already zero in the previous layer could be skipped, the calculations could be optimized. Although real-world examples will be more complicated, each layer could approximately be seen as having a probability of a point being non-zero.

$$P_n = \frac{\text{amount of coordinates with non-zero values}}{\text{amount of coordinates}}$$

As an example, we had 3 layers with the values $P_0 = 90\%$, $P_1 = 5\%$, and $P_2 = 50\%$, and started with $200 \times 200 = 40000$ coordinates. Below, the number of total calculations that were done is calculated, with the only difference being the order of the layers.

$$\{P_0, P_2, P_1\} = 40000 + 40000 \cdot 0.9 + 36000 \cdot 0.5 = 40000 + 36000 + 18000 = 94000$$

Figure 3.5: An example of a suboptimal execution order

$$\{P_1, P_2, P_0\} = 40000 + 40000 \cdot 0.05 + 2000 \cdot 0.5 = 40000 + 2000 + 1000 = 43000$$

Figure 3.6: An example of the optimal execution order

As we can see, the second order 3.6 of layers resulted in a significantly lower number of calculations compared to 3.5, which resulted in a faster analysis and therefore a better user experience by enabling better real-time feedback.

3.8 Gathering of Data Used in Analysis

The following section contains the methods used for retrieving and calculating some of the necessary information that is used in the analysis worker.

3.8.1 Speed Calculator

In order to be able to perform the necessary calculations for the heatmap, the speed of the tracked animal was needed. This was because the calculations were based on how far the animal could move within a time period. For the Atlantic sturgeon, the experts that were consulted did not have a value for the maximum speed or average speed of the sturgeon. Although there were some values to be found online, the scientists were not sure that those would apply to the sturgeons in Gothenburg. That was instead something they would investigate when the sturgeons had settled in Göta Älv. To cope with this discrepancy in the information that was needed for the calculations and what was available, the chosen solution was to create a tool in the program that calculated the approximate speed of the sturgeons based on the data that was uploaded, essentially creating a bell curve of speeds.

What was calculated was the mean, median, 90th percentile ("maximum speed") and 10th percentile of the speeds that the individual sturgeons had on average. In order to accomplish this various calculations were executed, such as converting the differences in latitude and longitude into distances in meters, and converting the times to absolute times including the time offsets that were given in the data, and performing the statistical calculations. Of course, this entailed using assumptions in order to be able to do anything at all. For example, the fish was implicitly assumed to be swimming in the general direction from one sensor to the next one it was detected at, and the speed statistics that resulted from this were technically "mean and maximum speeds of fish getting from one sensor to the next, independently of how their exact swim path went in between them".

Another thing that was done was creating an option to take the speed of the river water into account (subtracting it from the instances where the sturgeon was swimming in the direction of the flow and adding it to the instances where the sturgeon was swimming against the flow, to isolate the speed of the fish itself). As there was room for improvement in the initial solution, this feature was created as an option and could be completely ignored if desirable.

3.8.2 Cutout Heatmap over Water

The Sturgeon is a fish, which means that the heatmap should only be displayed over water. This was accomplished by firstly retrieving map data of water from

Open Street Map (OSM) and secondly converting this data into a desirable form and creating relevant functions.

Uploading data from sensors
The specific implementation created a more user-friendly upload functionality and made the user interface easier to use since the only thing needed to be done was to choose three files and nothing more. This creates an opening for a wide range of users from different backgrounds; this range is all from teachers, scientists, general public, to more technically suited people like data scientists, data engineers, and people who use and create software on a day-to-day basis. By using a mechanism called data management pipelining, we could create a program that enhances user-friendliness in the user interface for uploading data; without this mechanism, the data uploading procedure would be more technically advanced and complex for the end-user.

Retrieving Water Data
The GeoJson file used in the "check water function" is fetched from OSM. It uses two different sources, but both are based on the OSM data. The first data source is a pre-processed file containing ocean data. The original data is not directly useful for seeing where water is as it is stored as splines and contains errors that need to be fixed before water intersection can be done [30]. The processed data is instead stored as polygons in a Shapefile [45]. The other source of data is for retrieving the inland water data. This is done by querying the OSM Overpass API using filters for only specific water sources [31]. The data returned from the Overpass API should contain "all inland bodies of water, both naturally occurring & man made." [31] This data combined with the ocean polygons should contain all bodies of water around the globe.

Check Water Function
The essential function to make a cutout in the heatmap was to be able to check if a certain coordinate contained water. The retrieved data from OSM was a Geojson file containing Shapely polygons. Using the library Rasterizeio, the polygons could be converted into a raster (2x2 matrix containing ones and zeroes depending on water or land). To check if a coordinate contained water, a transform was created, relating a certain coordinate to a value in the matrix. A transform was needed as the conversion from the spherical earth to a flat map is not a linear operation. The transform returned one if the map contained water and zero if it was land. Using this function, a cutout of the heatmap over water could be achieved. When the heatmap was generated over the same area, the water data was only needed once. This also meant that the rasterization was only done once. Therefore, caching of the raster sped up this operation considerably when used over the same area multiple times.

3.9 User Interface

The following paragraphs will describe how the user interface was created, followed by how it was tested.

3.9.1 How the Mockup was Made

The first step of designing the user interface was to create a mock-up in Figma of what the final design could look like. This was done so that the group could get a rough idea of what features were to be prioritized and which were less important. When designing the site, a lot of inspiration was taken from other websites with similar visualization solutions. The first one was Strava [38], which uses heatmaps to visualize GPS data. The second was the SMHI radar tool [34], which combines a heatmap solution with a timeline. The third was Livelox [24], which uses a side panel to determine which GPS data should be shown. The following mock-up was created:

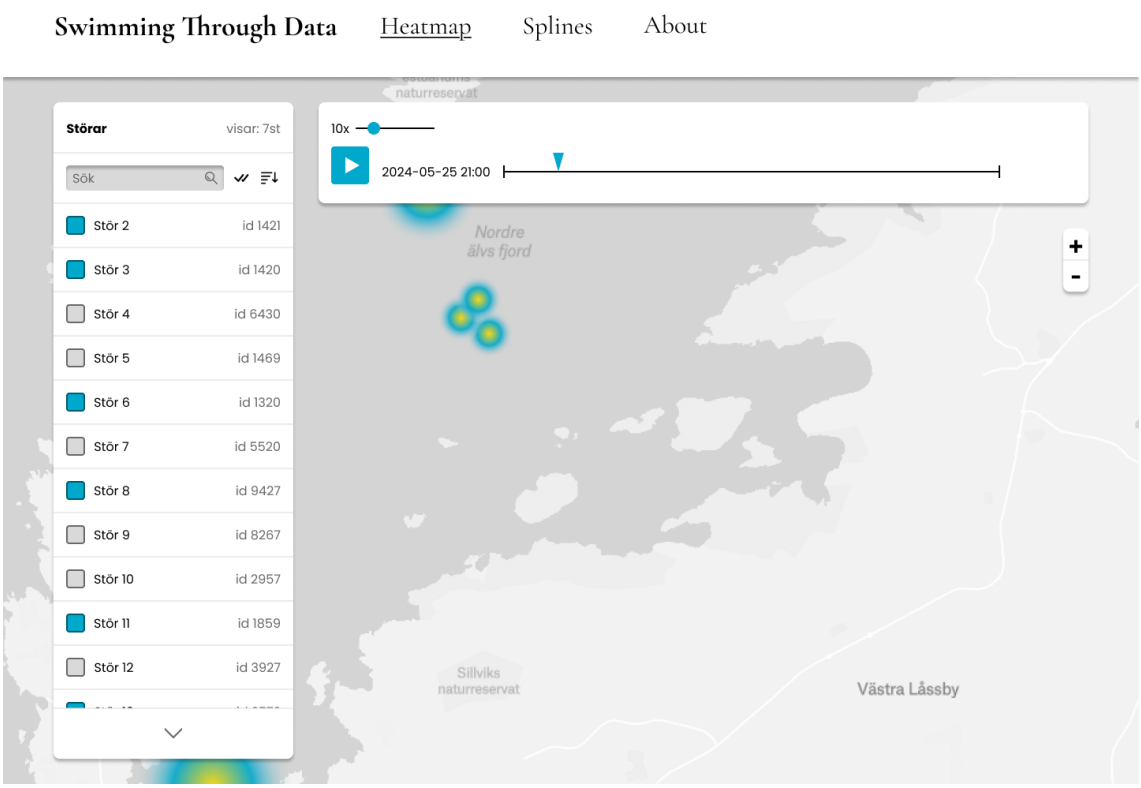


Figure 3.7: Figma mockup of the user interface.

After feedback from the project's supervisor and group members, it was decided that the main focus of the application should be on the interactive map. Therefore, menus were redesigned into drop-down menus that took up less space.

3.9.2 How the Frontend was Made

It was decided that the user interface would be coded in React, due to its efficiency in updating visuals. By using the mockup design from Figma as a template, a graphical user interface was created. Initially, it had no connection to the backend, meaning that buttons, text fields, etc., had no functionality. Three different front-

end pages were coded with the goal of fulfilling the main purpose of the project, as well as the sub purposes. The pages were:

1. **The 'homepage'**, which is the actual visualization tool with the interactive map.
2. **'Add data'**, with the aim of having an easy-to-use interface for uploading data.
3. **'Data'**, with the purpose of highlighting the new format of uploaded data.

The interactive map was coded using Leaflet [20], an external library. "Leaflet.heat" [21], a plugin for Leaflet, was added to create the heatmap overlay.

To make sure that the visualization and the overall user experience would become as clear as possible for popular science audiences, three different accessibility features were added. Tooltips, user feedback, and heatmap settings that let the user adjust the playback speed and the color of the heatmap.

3.10 User Tests

To better understand the performance and user experience of the visualization and the program, user tests were performed. By combining insights from qualitative, A/B, and heuristic tests, deeper insights could be achieved. This worked toward the goal of a better user experience by enabling the improvement of the program based on real user insights. The following section explains how these three tests were used to get different insights on the performance.

3.10.1 Qualitative Test

A qualitative test was performed on 6 people representing the popular science audience. A test protocol was created with the intention of testing Learnability, Efficiency, Memorability, Errors, and Satisfaction, see appendix D. The test as a whole was semi-moderated, and after doing tasks, the subjects were asked questions. In general, the results were more based on the actual performance than what the subjects said posterior to completing the tasks, but both were weighed in.

The test was divided into different sections, starting with a short introduction where the participants were told to think out loud during the test so that it was easier to analyze the thought process of the users. Then they were given various tasks that simulated what you would do in a normal workflow. In order to test the learnability of the program, the subjects were not given any specific instructions on how to do anything, just what the goals were, and were left to figure out how to do it from the program itself. The tasks were:

- Uploading data
- Navigating to the heatmap tab and choosing a specific time interval

- Studying the heatmap
- Changing the color theme
- Changing the playback speed

In order to test efficiency, all of the tasks were timed. This also helped to test memorability, as the subjects were asked to repeat all tasks after taking a break, where they were timed once again to see if any improvement in speed was made, which would be one indicator of a good level of memorability. Satisfaction was evaluated by asking follow-up questions to the subjects about their experience.

At the end of the test, the subjects were given some time to try whatever they wanted and were encouraged to actively try to crash the program. This was a way to try to catch errors that might otherwise go unnoticed.

3.10.2 Heuristic Test

Early in the development, heuristic tests were used to guide the design process by interviewing experienced people working in the area. A first heuristic interview took place on the 11th of February with Ichthyologist Dan Calderon. He started by giving information about the background of the Sturgeon as well as the project and his involvement. He also gave information about how the sensors work and what the possible sources of error are. For example, an important source of error is if a tracked Sturgeon gets eaten by a larger fish, which would mean that the sensor would reside inside another fish than the intended and therefore give false sensor data. This information was essential to get a deeper knowledge about what the fish tracking data consisted of and how it was generated. The heuristic test helped shape the purpose of this project as well as provide a basic understanding of other involved scientists' problems, which could be used to later create a MoSCoW analysis.

A second heuristic test was performed on the 20th of February with Dan Calderon, Susan Gotensparre, and Johan Höjesjö (see Appendix A). With their broader combined competence, even more specific questions could be answered. Höjesjö could provide opinions about the analytical usability of the program for Ichthyologists, Gotensparre provided information about what the target audience was, and by experience talked about how this kind of project usually is communicated and what to think about. After the interview, a MoSCoW analysis could be compiled using the different inputs they provided.

4

Results

4.1 Improved UX by Data Pipeline Efficiency

The data pipeline works as expected, the user experience did improve significantly since the user only needs to highlight the files and submit them. During the user tests, we did not receive any complaints about the layout and how the user interface looked. Because of all the parallel computations and the optimized code, we managed to build an optimized data uploading layout with only the "Must have" visual components.

4.2 Backend Architecture

The backend of the application was constructed around a modular and scalable architecture. The primary communication with the frontend and third parties was through a RESTful API that adhered to the stateless communication principle. Each request contained all the necessary context, allowing the server to handle it independently without retaining any session state. This improved scalability, simplified deployment, and enabled straightforward load balancing.

The backend performed several core functions and acted as a central communication layer between all system components. These core responsibilities included data preprocessing and transformation, data validation, job dispatching, persistent storage, and serving structured data via its RESTful API. Each of these responsibilities was implemented in a modular fashion, promoting a clear separation of concerns and abstracting away low-level implementation details from other parts of the system. This architecture improved maintainability, encouraged the reusability of components, and simplified future development or integration of new features.

In addition to handling asynchronous REST API requests, the backend was made to support background processing of work. Computationally intensive or time-consuming tasks, currently geospatial analysis, were offloaded to a dedicated worker pool. This approach ensured responsiveness in the API and prevented blocking operations from delaying other users or services.

The dispatching mechanism for these background tasks was implemented using Redis Streams. When a REST API endpoint wanted to schedule an analysis job, it pushed

a job description to a Redis Stream. A separate set of worker processes, minimum of one, continuously listened for new jobs on the stream. To ensure mutual exclusion of jobs between workers, a PostgreSQL table was used to manage metadata, processing state, and retries if a worker fails, etc. The hybrid coordination between Redis for fast real-time message queuing and PostgreSQL for durable storage and job accountability ensured if a job should be done, then it would be done once even if there are failures.

The design supported horizontal scaling, allowing multiple workers to process jobs in parallel while avoiding duplication or race conditions. The architecture was thus well-suited for batch-oriented workflows and on-demand computations.

The system was designed to be extendable in future iterations. This meant that even though the system currently was unauthenticated and unsecured for simplicity, the architecture allowed for easy integration of such features.

4.3 Analysis

The analysis pipeline worked as intended, although it was limited due to the short development period. This section will further analyze our results.

4.3.1 Overall performance

Some images were generated to display the resulting heatmap directly from the analysis pipeline.

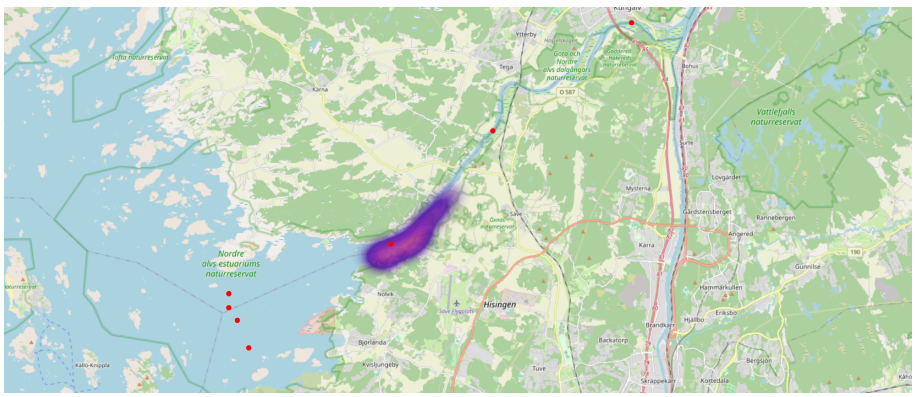


Figure 4.1: A screenshot from the analyze pipeline generated by using faked time data, but real sensor positions.

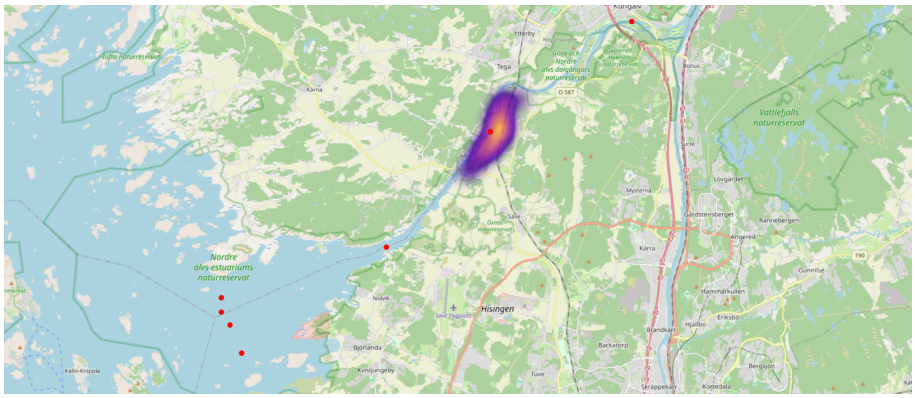


Figure 4.2: The same sturgeon but a later time compared to 4.1.

The resulting heatmap made it easy to see where a specific sturgeon was located at a specific time. Moreover, since it was by design a heatmap that was the output, it was easily understood that the sturgeon could be located anywhere within the marked area.

To better understand if the results were reasonable, some simple checks were done. By looking at the time data, it was found that the nearest sensor detection for that frame was indeed the same sensor as the nearest sensor on the heatmap! It could also be seen that no points were generated on land, which indicated that the water detection part of the pipeline worked as intended.

By comparing 4.1 and 4.2, the sturgeon did indeed move as expected! This further verified that the analysis pipeline was working as intended.

4.4 User Interface

The final product consists of three different pages: the homepage, add data, and data. The following paragraphs explain the purpose of each page and which accessibility features were implemented.

4.4.1 The homepage

The homepage is the first page the user sees when opening the website. They can navigate to it by pressing the logo, “Swimming through data”. The page has an interactive map with common navigation features such as pan and zoom. There are several buttons on top of the map that have been coded as drop-down menus. Below follows a list of implemented features. An overview of the homepage can be seen in figure 4.3.

- **Id list:** A drop-down menu with a list of all the sturgeons. The user can choose up to one sturgeon at a time. The "active" sturgeon is the sturgeon for which data will be retrieved.

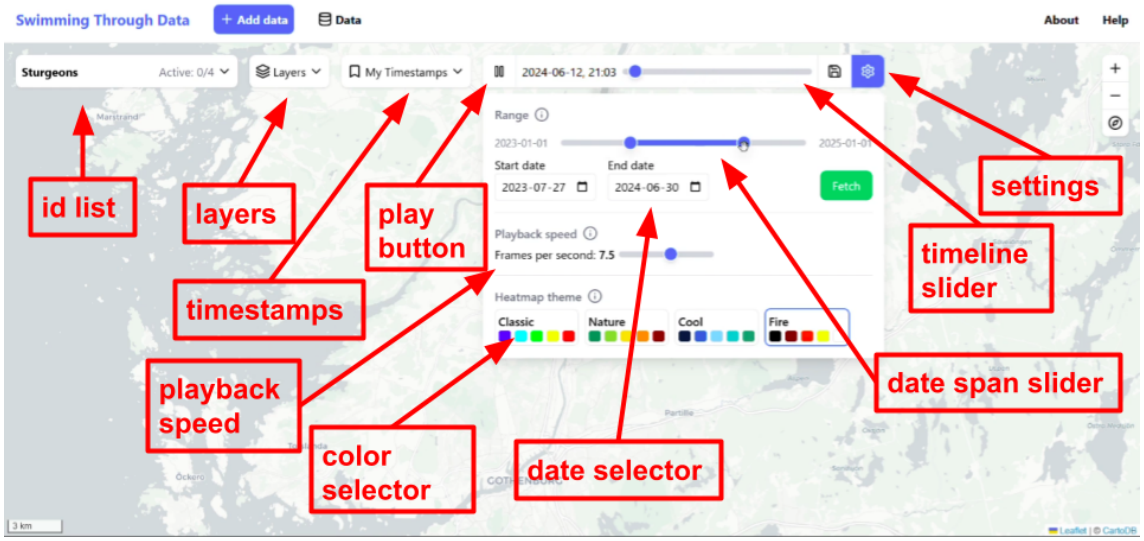


Figure 4.3: Application front end with labels.

- **Time slider:** The time slider has a knob that can be moved horizontal. When moving the knob, the frame to be displayed will be changed. The date left of the slider indicates the timestamp of the current displayed data.

- **Play button:** The play button can be pressed to automatically move the time slider. The slider moves at a certain speed, which can be determined in the settings.

- **Settings:** The settings button opens up a drop-down menu with settings for the heatmap.

- **Date span slider:** To display heatmaps, the user has to fetch heatmaps from the backend. This is done by selecting a range of dates, this can be done by tweaking the two knobs on the slider. After choosing two valid dates, the 'fetch' button can be pressed. This button press tells the backend to calculate new heatmap frames and send them to the frontend. This request could take a while. Meanwhile, the user gets messages about when the program is loading and whether the request was successful or not.

- **Date selector:** The date can also be selected by manually typing a date, or by clicking the calendar icon, which opens up a small GUI for a calendar.

- **Playback speed:** The speed of playback can be changed if desired. The default value is set to five frames per seconds, but the range goes all the way up to 30 frames per seconds.

- **Color selector:** The color of the heatmap can be changed by choosing one of the four color themes.

4.4.2 Add data

The “Add data” page 4.4 is the page where data can be uploaded to the database. Data can be uploaded by either pressing “Browse” or by dropping files into the dotted box. Excel, CSV, and JSON are the file formats that the program accepts. Uploaded files will appear on the right. Once at least three files are uploaded, the “Submit” button can be pressed. After pressing the “Submit” button, the program will load for a few seconds and then return either a green success message or a red error message.

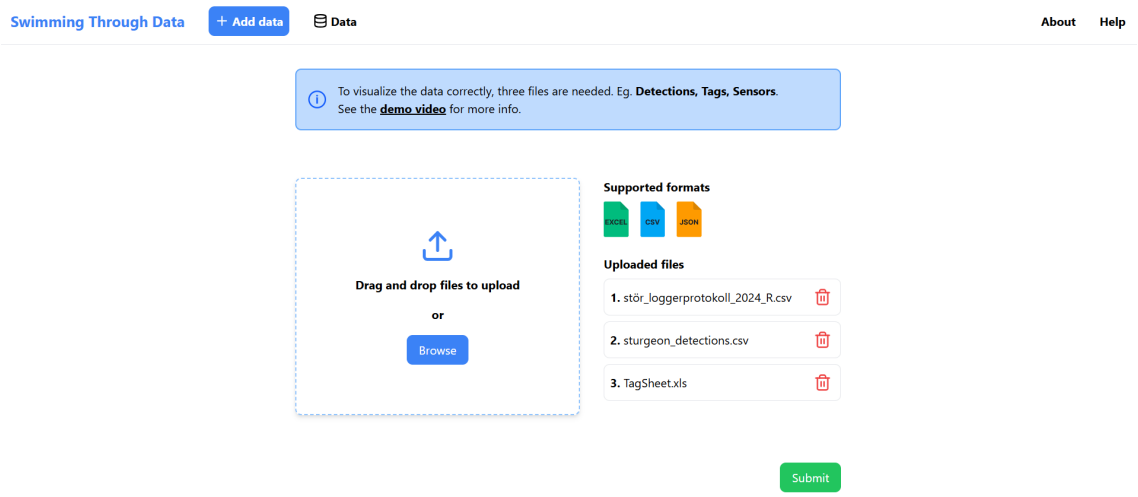


Figure 4.4: User interface for uploading data to the database.

4.4.3 Data

The “Data” tab’s 4.5 purpose is to view the uploaded data in its new format. This tab has no functionality; for example, the user can not edit elements. The data will come in several different tables; the user can swap which table to display by clicking the respective table name at the top.

4.4.4 Accessibility features

Some accessibility features have been implemented. These features aim to make it as clear as possible for popular science audiences to use and understand the program, which was a part of the main purpose of this project. How well these features help can be read in 4.5 User Test Results.

Tooltips: Tooltips can be found here and there; they are displayed as a small black box with white text. They are shown while hovering over certain elements that need further explanation. The idea is to help the user without sacrificing the workflow for the experienced user.

User feedback: The user is notified after certain events have occurred. For example, when the program is loading, when an error occurs, or when something goes

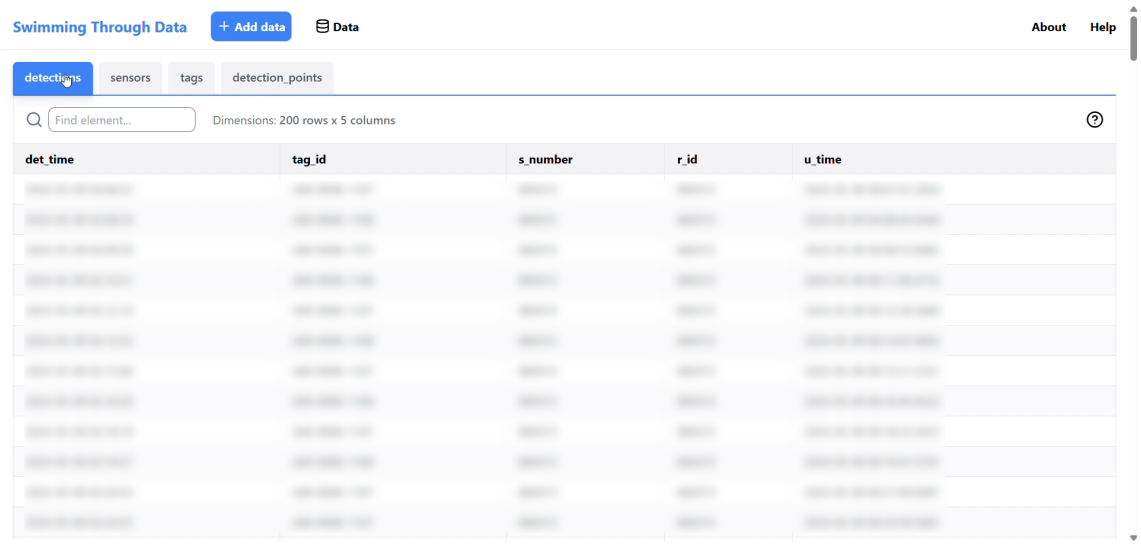


Figure 4.5: User interface for viewing uploaded data. The values have been blurred.

successfully. This is displayed in the bottom right corner as a color-coded box with a message. Red means error, yellow means loading, and green means success.

Heatmap settings: The playback speed of the heatmap visualization can be adjusted, allowing for a more precise and personalized visualization. The theme selection allows for color-blind-friendly options and options with more contrast.

4.4.5 What features were not implemented?

- **My Timestamps**: "My Timestamps" was supposed to be a feature that allows the user to save certain timestamps to the database. By doing this, the user would be able to go back to events that are considered important.
- **Layers**: The idea of "Layers" was to let the user choose which overlays to display on the map. The two default overlays are "Heatmap" and "Sensors". By clicking the checkbox on the left, the layer becomes visible or invisible.
- **About page**: The idea of the about page was to give the user a short introduction to the background behind the project as well as credits. It would be very similar to the project's GitHub repo.
- **Help page**: The idea of the help page was to give the user a deeper understanding of how the program works. This would be achieved by showcasing demo videos of how different features work.

4.5 User Tests Results

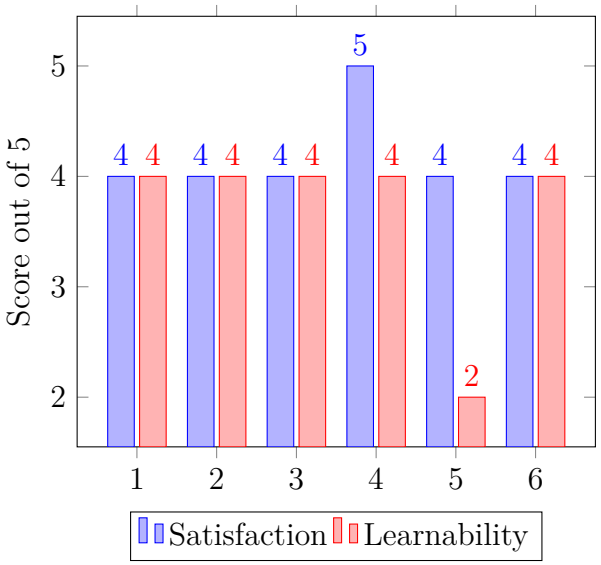
Six test subjects were used in the user tests, with the following demographics.

Subject Nr	Age	Gender
1	22	Male
2	22	Male
3	23	Male
4	25	Female
5	22	Female
6	22	Female

Table 4.1: Demographics of test subjects

4.5.1 Subjective Evaluations - Satisfaction & Learnability

The outcome of asking the test subjects 1-6 to rate their overall satisfaction with the program and the learnability of the program out of five is shown in the bar chart below.



The overall satisfaction with the program was on the high end of the scale, and there were a few different specific comments motivating this and also why it did not reach a full 5/5 for most. They were

- It was good that the user is not bombarded with too much information
- The popup saying the data has been uploaded gives the user a clear understanding
- The simulation was smooth to run
- It was a good feature to be able to play the simulation faster or slower depending on what you are looking for
- It was good that color blind people were considered in being able to change color themes

- A preference that the settings button and the timeline slider should be separate units instead of being totally connected

- The simulation itself should be more accurate and not show fish possibly being in lakes not connected to the river (this was however not supposed to be the end result, just a consequence of a part of the visualization algorithm not being implemented when the test was run)

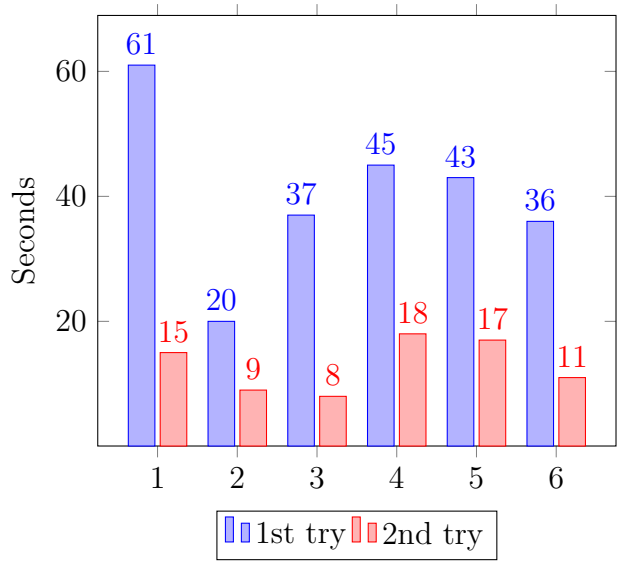
- A preference for a Swedish rather than American convention for typing in dates

The practically unanimous motivation for the learnability score was that it was not so explicit to understand that you had to go to specifically the settings button to choose the time interval after uploading the data, but that it was very clear how to do everything once you had opened the settings. It was also generally stated that although it was a bit confusing the first time, it was entirely possible to try things out until you found the right button. One test subject suggested simply adding a popup after you've uploaded your data that points to the settings button and tells you that it is there you can find the time interval setting, and that if that were implemented he would consider the learnability score to be a 5/5. One test subject also had a comment about the label of the speed slider being "frames per second", which he expected to be related to smoothness rather than speed, and suggested renaming it as "playback speed" or something similar in order to make the settings more intuitive which could give them an even better learnability.

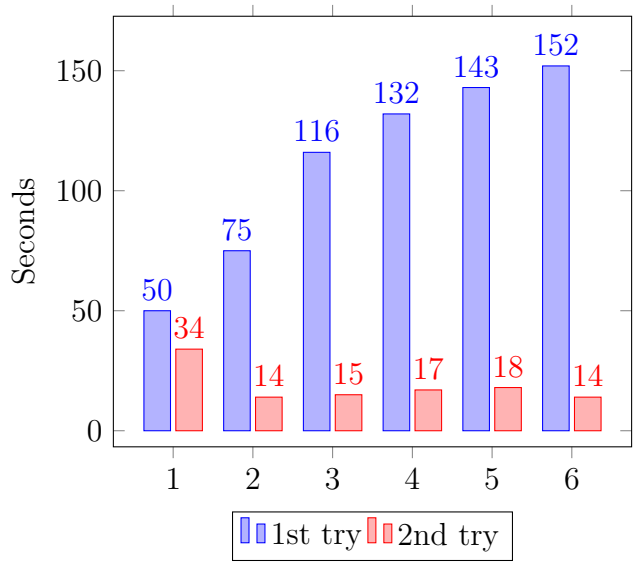
4.5.2 Memorability

For the memorability, two main tasks were timed, as the other tasks (such as changing the color theme) practically took 1 second for all subjects to do even the first time, so there was not much to measure for those other tasks. The first timed task was to upload data files from the computer to the program. The second timed task was to select a time interval between two specific times to visualize the uploaded data between.

The times it took the test subjects 1-6 to complete the first task, i.e. upload data, are shown in the bar chart below.



The times it took the test subjects 1-6 to complete the second task, i.e. choose the time interval, are shown in the bar chart below.



Noteworthy is that the second time for test subject 1 included a crash of the program.

4.5.3 Errors

A few of the subjects (1, 2, 3, and 4) managed to cause errors in the program. Subject 1 and 3 got a crash from typing in the date wrong when selecting the time interval. Subject 2 pressed “clear” for the input of the time interval, which made the time interval disappear and no longer be defined, causing the program to crash. Subject 4 went into the tab labeled “Data,” and it showed “failed to load data” - this was, however, not a real error but just a consequence of it not being implemented yet. Subject 4 later had a problem that after starting to type in the date and then switching to the calendar interface for choosing the dates, it was not possible to click on the interface (the mouse was still in typing mode). Subject 5 and 6 had no

errors.

4.5.4 Efficiency

The average time it took the test subjects to perform task 1 (uploading data) was 13 seconds. Similarly, the average time it took the test subjects to perform task 2 (choosing a time interval to visualize the data on) was 19 seconds. This shows that task 2 was a slightly less efficient task in the program.

A flaw for the efficiency of the data uploading that was pointed out by a few test subjects was that they were not able to mark multiple files with ctrl and upload them simultaneously that way (see Appendix B). Something that multiple test subjects reacted to when choosing the time interval was that the time used the American format [44] which caused them to type in the numbers in the wrong order and requiring them to re-type. They suggested using the Swedish time format [43] to avoid this confusion and make it easy to type from "muscle memory" without thinking too much.

5

Discussion

This discussion elaborates on several key findings stemming from the research conducted in this thesis. It addresses the mechanisms of data transfer between system nodes and the potential data integrity challenges arising from sensor mobility. Furthermore, it examines the complexities inherent in implementing certain technical features and their consequent impact on end-user interface design. The discussion also covers the novel application of techniques adapted from unrelated software domains, alongside the introduction of mathematical algorithms for robust data analysis within the developed program.

5.1 Comparison to other visualization methods

From the literature study, it was found that a variety of methods of visualizing spatial data exist. This project had a large emphasis on not misrepresenting data.

The raw data methods have the positive aspects that they do not misrepresent data as it uses it in raw form. However, the simplicity reduces the clarity of the visualizations as not many laypersons can imagine the real paths of the fish from it, and as mentioned in the introduction, this project intends to create visualizations that are easily understandable for non-scientists.

The linear- and nonlinear interpolation methods help the user see more clearly where the fish might have been, but at the cost of misrepresenting data as the data points are interpolated. This also creates artifacts such as paths crossing land. Also, if the meaning of interpolation is not communicated to the user in the right way, they might not know that the fish do not actually follow the path, but can only be approximated to be there. These problems are what the third method of the literature study tries to fix.

The Dynamical Brownian Bridges Movement Models do not misrepresent data as it only visualizes the probability of the fish being in a certain area. It is also an easy to understand visualization as the strong colors in the center imply that the fish is more likely to be there and have a fading chance of being farther away from the center. It is also built on the statistical model Brownian Random Walk, which is useful for analyzing data. However, the disadvantage of this approach is that it uses a model of the fish where a C-value determines how likely the animal strays

from the direct path. This value is essential to use this model in a realistic way, and as the Sturgeon is a newly introduced fish that has not been studied in depth yet, there are multiple missing pieces to use this model here.

Our approach has the same positive aspects as the dBBMM model as it does not misrepresent data and is clear to understand. The difference is that our model only uses the average speed of the fish; no C-value is needed. Which means that it can be used when the studied species have not gotten to the point where it is known how it circulates and moves in water. The disadvantage is that our model is not built upon a statistical model and therefore has less analytical qualities. Nevertheless, it is a useful model to get an overview of the fish movement over time, and you know that data is not misrepresented, which means scientists can draw their own conclusions and see new possible movement patterns.

Applying the dBBMM to sturgeon data is also not straightforward, as the model is primarily designed to represent movement between two defined locations, whereas our approach aims to model the sturgeon's movement within a complex network of interconnected waterways. We believe it is reasonable to view our model as complementary to existing approaches, each being suited to different ecological contexts and data availability.

5.2 Demographics - Test Subjects and Target Audience

The user tests are meant to represent the popular science audience for the program and visualization. As seen in the results, all six test subjects were in the age range of 22-25 years. This is, of course, not representative of the ages of the entire population. When trying to apply the results to the whole target audience, it should be taken into account that the age group 20-24 has the second highest computer literacy of all age groups (beaten only by the age group 15-19), and the computer literacy rate goes down by about 10 points per age group after that [8]. This and other factors mean that although the results were positive, they are limited as we did not have a representative test pool. Still, as long as this is kept in mind, the results are valuable and provide far more insights than not having any independent users test the program at all. For the visualization part, we considered the most important thing to be that the test subjects had no previous knowledge about our project, and this was fulfilled.

The part we have identified as possibly more problematic is that the test subjects, being in this age range, were all students. This might mean that they were more used to (and having it as a present practice) interpreting different types of data representations than other demographic groups. Taking this into account, we should estimate the visualization to be less understandable for the general user than it was for our test subjects. It was already evident from the user tests with our test subjects that some short clarifying text about how the visualization works and what it represents would make a difference in the understanding of it, and taking into

account the whole target audience this may be even more important to implement. We also expect that with some explanation, the heatmap will be possible to interpret for the general user, especially as heatmaps are already used in day-to-day things like weather apps, so they may be quite familiar.

5.3 Applying User Test Information

The user tests provided useful information about what part of the UI could be made more efficient and clear. Unfortunately, the tests were conducted late in the development as we needed a working application to be tested. This gave no time in the project timeline to iterate and improve the UI design. Many of the things mentioned in the tests, such as finding the settings button (see appendix E), were things we already knew should be designed in a more efficient manner, although the lack of time left these changes underprioritized. However, the average satisfaction score tells us that the overall function and design of the UI work well. To get a more representative user test response, it could be useful to select people of a wider variety of ages as the purpose is to present the data to a general public.

One pattern that could be seen in the qualitative test was that the male subjects generally were faster at navigating to the settings for choosing a time span to visualize the data on, and had a lower average rating of learnability. Since it was a qualitative test, the sample size is too small to draw any such quantitative conclusions from the data. However, it may be interesting to investigate whether this holds true for a larger population in further research and development of this program, since the majority of the developer team actually was male. If there turns out to be any such bias, it should be rectified to ensure the user experience is good for the whole target audience.

The results on memorability show that the second time a user comes back to the application, it is much faster, meaning it is quite effortless to memorize the workflow and where to find things. This combined with the learnability score (which on average was 3.67 out of 5.00) means that while it is slightly unintuitive the first time using the app, although still possible to learn independently, the second time using it is much easier. This is good for the scientists that would probably use the program many times to generate visualizations to show. For the popular science audience, the visualization itself is the most important part as they may not always have to interact with the program much on their own; however, if there are events where they do, their experience as first-time users can be improved by adding more popups or having an explanatory text next to the computer with the demo.

5.4 A/B test

During the meeting with the scientists, they showed a current animation of fish tracking data. This could be compared to our visualization using A/B testing to get information about how well ours performs. In that case, a quantifiable number would be needed to decide which version is the most clear design. When comparing

them, the function to optimize would be clarity, which means how easy it would be for the user to understand the visualization. From the theory section, we could see that an A/B test would need around 100 participants to get a reliable result. Therefore, a digital survey would be preferable. The survey could consist of a short explanation of the project, the scientists, and our visualization side by side and one question asking which visualization is easiest to understand on a scale of one to ten. After that, we could calculate the average score of the two visualizations, and if ours got a higher score, it would mean that it was a better design in regard to clarity, which was also this project's main purpose.

5.5 Future Development

During the project's duration, we gathered information on how the project could be improved in the future. This includes improvements to the backend, frontend, and to the analysis.

5.5.1 Data Management Development

The data uploading procedure is still very bare bones; it is not as generalized as we wanted. The amount of different data structures that can be applied in this program is very versatile, but because of time restrictions, resource restrictions, and other factors, we did not have the time to create an even more general solution. We have created a mechanism to solve this issue, but this creates a more complex data uploading pipeline where the user has to provide more information regarding the data and columns therein. This could lead to confusion and a more complex mechanism that could decrease user-friendliness. This limits the program to specific users that can and cannot use the application. It creates a more specific scenario where the user needs to be more technically experienced, which decreases the number of people who can use our application.

A solution for this problem could be to expand the regexes that are used. But this requires a lot of manual work and will, for example, not automatically work with new ways of labeling data or spelling errors. Another solution could be Named Entity Recognition (NER), as "it's particularly useful for quickly extracting key information from large amounts of data because it automates the extraction process" [15].

5.5.2 Artificial Intelligence Development

Artificial intelligence technology may improve the column classification mechanism since the mathematical probability theory would increase the hard-coded vocabulary by a large margin. This would improve long-term user experience because of the optimized classification mechanism in the pipeline, which increases the scale of different types of data structures that could be used. However, AI technology could cause some problems if not implemented correctly [1].

Researchers and scientists have realized a big problem: the reason behind the ma-

jority of AI scandals that occur in the world today is something called "BIAS" [28]. A bias is a standard behavior in a mathematical formula that could lead to falsely provided information, for example, gender, racial, or socioeconomic false painted perspectives. This can occur in large LLMs such as ChatGPT but also in other AI networks [37]. This bias could lead to false visualization of data, application crashes, falsely analyzed conclusions, etc. The solution for improvement of this so-called "BIAS" factor is slow and difficult. You have to collect diverse data, balance class distributions, use synthetic data to fill in the gaps for the missing data (artificially created data), etc. [33]. We realized that these limitations and types of technology are too advanced for the majority of us, and the time needed is something we did not have.

Another problem that we had was how resource-dependent the training process is. A good example is how difficult it is to train a generative pretrained transformer, which could require multiple graphics cards connected through very expensive computers [39]. The problem here is that the amount of data needed to train this AI model requires a powerful computer. This would decrease the diversity of users, and the user interface friendliness would decrease.

An artificial intelligence neural network creates a mathematical function that takes a specific input and returns another output. The problem with this approach is that the mathematical equation is based on probability theory and does not have a direct linear relationship, which can result in multiple different outputs [22]. This creates uncertainty. The problem is only solvable with a large amount of data, larger and more complex neural networks, and more powerful hardware-related resources [22].

The end-user is probably going to use this program on their laptop or a normal computer, which is not suited for artificial intelligence matrix multiplications because of limitations on the graphics cards. This decreases the diversity of end-users because of the cost-complex situation, the technical expertise, and how neural networks work.

5.5.3 Information view

The Atlantic sturgeon has impressive features and an interesting history, which could spark interest and raise awareness about human impacts on ecology to the popular scientific audience if it were incorporated into the program. This aligns with the purposes of this project. It was not done in this first edition of the program because it was not absolutely necessary and because we tried to make it as general as possible. Combining specialized information with a general and adaptive application would have needed more thought and work, which would be nice to do in the future instead.

5.5.4 Speed calculator generalization

For the sturgeon dataset, there were problems with the data showing impossible time spans between sensors at times (e.g. swimming at 110 m/s), so outliers had to be removed. The outliers (or, completely unreasonable values) were removed in a "hard-coded" way, removing everything above a chosen threshold for the Göta Älv

sturgeon dataset in order to get results that were as good as possible. However, some other statistical methods for removing outliers were looked into and a method for using a z-score test was implemented. However, the results of this were not quite as satisfying for this particular dataset as removing outliers in the less general way. Further development could be done in this area, finding an optimal and general statistical method for removing outliers in this type of application.

There are many potential upgrades that could, and probably should, be made to the "taking river water speed into account" feature of the speed calculator in order for it to be totally accurate. Some issues are that the water flow is the data that is usually available, and that means the speed of the water varies over the river depending on how wide and deep it is in different parts. In the first implementation, the speed is a constant mean approximation of the speed based on typical values of depth and width for Göta Älv and approximating its shape as the cross-section being a half ellipse. The whole thing should also be generalized for any dataset.

6

Conclusion

This project culminated in the creation of "Swimming Through Data", an open-source and interactive tool designed to visualize geo-spatial tracking data, with a particular focus on the movements of aquatic creatures such as the Atlantic sturgeon. By interpolating and interpreting the sparse data typically gathered by acoustic sensors, the project's core mission to generate clear heatmaps has been successful. All while ensuring the integrity of the underlying data.

A key success of this endeavor was the development of a fully functional tool capable of data uploading, processing, sophisticated analysis, and the generation of dynamic, interactive heatmaps. The project successfully tackled the issues surrounding the refinement of raw sensor data, transforming it into a more readily usable and accessible format. This data processing and analysis is supported by a durable and extensible software architecture. This includes a backend equipped with a RESTful API and a data management pipeline, alongside multiple analysis workers to handle computations. The user-facing frontend was meticulously crafted using React, with Leaflet and Leaflet.heat, to deliver engaging and interactive map displays. A user-centered design process, incorporating the Triple Diamond approach and MOSCOW analysis, guided the development.

Feedback from user testing was largely positive, highlighting overall satisfaction and a good degree of learnability, though some suggestions for user interface improvements were noted. Crucially, the chosen visualization method proved to be both clear and effective in representing the often-encountered sparse datasets.

This project, "Swimming Through Data", successfully delivered a tool for visualizing geo-spatial tracking data, focusing on aquatic animals like the Atlantic sturgeon. The main goal of creating clear heatmaps from sparse acoustic sensor data, without misrepresenting the information, was achieved.

The project resulted in a functional tool capable of data uploading, processing, analysis, and interactive heatmap generation. It addressed the challenge of refining raw sensor data into a more usable format. This is supported by a durable and extensible software architecture, featuring a backend with a RESTful API, a data management pipeline, and analysis workers. The frontend, built with React, Leaflet, and Leaflet.heat, provides interactive map displays. The development process followed a user-centered design, including the Triple Diamond approach and MoSCoW

analysis. User testing indicated high satisfaction and good learnability, along with suggestions for UI improvements. The visualization method proved effective for representing sparse datasets clearly.

A

Interview with an Expert Panel

This is a summary of an interview with an expert panel conducted on 2025-02-20. The expert panel consisted of:

- Johan Höjesjö - Fish ecology scientist
- Dan Calderon - Creator of the Sturgeon project and scientist
- Susan Gotensparre - Visual communicator for Marine Biology projects

The following points were concluded, and the list of requirements is based on them.

- A use case is to quantify areas where fish play and other fish activities are common
- Also conclude if fish are in areas where scientists have hypothesized that they should be
- Also quantify the modality of fish i.e. the percentage of planted fish babies surviving
- Johan sees no use case in interactive animations for the purpose of science as they cannot include moving images in research papers
- Although, they see the use case of visualizing to the public
- The public consists of individuals as well as Municipals, County Council, and Media
- The program should be able to handle many similar projects' data, such as the eel and tuna outside the coast of Marsstrand
- The program should show where habitats and areas of interest are located
- By using different colors for different species you can use the program to see correlations between species which would be interesting for scientists
- Until now the scientists have not looked so much at the habitats of fish populations but they have recently started to do so, leading to them agreeing that there are a lot of use cases for our program

- The program needs to correctly classify extreme cases such as dead fish or faulty sensors
- The scientists' main use case is to scroll through large datasets and find probabilities and average time data in different regions
- The visualization is mainly used to communicate their findings to the public. This is done through, *Vetenskapsfestivalen*, *Västerhavsveckan*, *Forskarfredagen*, all types of *media*, walking path along the locks with QR codes for people to scan with their *smartphones* and also as *exhibitions* in museums such as Naturhistoriska. This means that there are two main ways to use the program for the public:
  - **Interactively** - where a scientist use the program together with people from the public or individuals use the program by them self
  - **Static** - where images and videos are generated to use for different purposes
- They said that media easily spreads news about sturgeons, perhaps because people of Gothenburg are interested in the history of the city and this is a cool kind of fish
- Dan and Susan often think about how to reach out to the public
- It would be interesting to, in the future, correlate areas of fishing with areas of fish activity
- The project needs to be developed to cope with a continuous data stream in the future as they maybe could attach antennas to the sensors above water, and also animals above water use GPS tracking data, which is live
- A use case that would save the scientists money is to be able to daily review the new data in an easy fashion
- An inspiration for a new fish tracking data standard is the European Telemetry Network Standard

B

MoSCoW Analysis

The following is a MoSCoW analysis that was conducted with three scientists involved in the project, see appendix A. The involved scientists were Johan Höjesjö - Fish ecology scientist, Dan Calderon - Creator of the Sturgeon project and fish scientist, as well as Susan Gotensparre - Visual communicator for Marine Biology projects.

Must Have

The front end should be designed in two ways where one is tailored towards fish tracking scientists and the other is tailored toward the public. It could be designed in one view where you have the option to hide or show the relevant parts, or in two separate views.

- Scientists view should be useful for use in fish science, which means clear display of data and labeling of what values represent (such as averages, probability density etc.). Also easily scroll through large datasets and adjust timeframes or other aspects that could help them quantifying hypothesises. The data needs to be displayed in a way that is not misrepresenting.
- Public view should be designed in a way that makes the data easy to understand for laymen such as citizens and people working for the municipal, county council or organizations. In this view it is acceptable to lose a bit of accurate data representation in favor of making the data understandable to non-scientists. This could include interpolating datapoints on splines.

Should Have

- Ability to download the visualizations as images and videos.
- Ability to add multiple datasets (such as other species or fishing boats) and visualize areas of intersection.

Could Have

- Ability to add custom layers to the map such as habitats, areas of interest or river ecology information.
- Beyond a probability heatmap there should be a predefined layer for visualizing

the path of fish using for example splines and interpolating between datapoints.

- Ability to detect extreme events such as dead fish or faulty sensors.
- Ability to communicate a connection between the Sturgeon project and ecological and cultural values relevant for the the county council of Västra Götaland.
- Ability to handle live tracking data.

Won't Have/Would Like

- Information about the Atlantic sturgeon and its history, as described in the theory section, accessible for the users to read.
- Detecting if a sturgeon has stopped moving for a certain amount of time and deeming it dead, removing from further analysis.

C

Explanation of communication between frontend and backend

While frontend development is responsible for the user-facing interface and experience, an application's smooth operation hinges on effective backend management. Therefore, backend development is a crucial component of the overall user experience, as its performance can either impede or enhance the application's speed.

C.1 What is an API?

The software components operate in isolation and need to retrieve data from, or instruct another application to perform one or more specific tasks. This necessity for interaction between different systems, potentially written in different programming languages and running on a singular or multiple distinct platforms, demands a well-defined mechanism for communication. An application programming interface, or in short API, serves as such a mechanism. It defines a set of rules and specifications that permit one piece of software to interact with another. Analogous to how functions within a programming library provide an interface for utilizing the library's functionality, an API provides an interface for interaction between programs.

C.2 What is REST?

Representational State Transfer or, in short, REST is an architectural style developed with the goal of improving the performance, scalability, simplicity, falsifiability, visibility, portability, and reliability of distributed hypermedia systems, primarily the World Wide Web [10].

REST is based on principles that leverage the decentralized and standardized functionalities of the web. The most central principles for a RESTFUL API include: [10].

- **Separation between Client-Server:** The client (in the invoking application, in this case programmed for example with React [36], using Typescript [27]) and Server (the providing application, for example programmed with

FastAPI [9], using python [11]) are separate concerns with distinct responsibilities. This allows for independent development.

- **Statelessness:** Each request from the client to the server must contain all the information for the server to understand and fulfill the requests. The server stores no client-specific session information between requests. This simplifies server scaling.
- **Cacheable:** If a response is cacheable, the client can reuse the response for future similar requests, improving performance and reducing server load.
- **Uniform Interface:** This is one of the most defining principles of REST and central to understanding how to consume a RESTFUL API. It consists of four parts:
- **Identification of Resources:** Each program that the API manages, for example a user management program, a product management program, a post management program that is treated as a resource. Resources are identified by a unique identifier, typically a URI (Uniform resource Identifier), which in practice is often a URL.
- **Manipulation of Resources through Representations:** When a client wishes to interact with a resource, this is done through a representation of the resource. A representation is a format that depicts the current state of the resource. Common formats include JSON and XML, both of which are structured and readable formats.
- **Self-descriptive Messages:** Each request and response contains enough information for the recipient to understand how it should be processed. This include the use of standardized HTTP methods and media types.

C.3 The communication flow in the RESTful API

Communication between a client and a server via a RESTFUL API occurs over the HTTP protocol. The flow can be described as a request-response cycle: [10].

Client's Request:

The client sends an HTTP request to the server. This request contains an HTTP method that indicates the type of operation that the client wants to perform on the resource. The most common, directly linked to basic data operations (CRUD - Create, Read, Update, Delete), are: [10].

- **GET:** Retrieve a representation of a resource (Read).
- **POST:** Create a new resource, often under a specific collection (Create).
- **PUT:** Update an existing resource, or create it if it does not exist (Update/Create).

- **DELETE:** Remove a specific resource (Delete).

The other components are:

- **URI:** The unique address of the resource
- **Headers:** Additional metadata about the request (what format the client accepts as a response, authentication information).
- **Body:** Contains a representation of the data to be created or updated, often in JSON format.

Server Processing:

The server receives the request, identifies the resource and method, and performs the requested operation. For example, retrieve data from a database, save new data. [10].

Server Response:

The server sends back an HTTP response to the client. This response contains:

- **Status Code:** A numerical code indicating the outcome of the request.
- **Headers:** Metadata about the response.
- **Body:** Contains the representation of the requested resource (for GET), or a confirmation/error message (for POST,PUT,DELETE), most commonly in JSON format.

D

User Test Protocoll

1 Introduction

Tell them to talk out loud about what they are thinking during tasks - “thinking out loud”

2 Tasks

We will time all separate tasks below

We will also take a screen recording to be able to look back

2.1 Load data from Excel files that are in a folder

Was there anything in particular that you would have liked to be different? Was it clear to understand when the data was fully loaded?

2.2 Choose a time between two dates (choose May 21st to May 22nd, 2024)

Was there anything in particular that you would have liked to be different? Was it easy to find how to do it? Would you have figured out that you could choose a time on your own without being given this task? Was it clear what the start and end dates were?

2.3 Show heatmaps

Could you find the fetch button?

2.4 Look at the heatmap

Was there anything in particular that you would have liked to be different? What happened/what did the fish do? Do you understand why heatmaps are used to visualize this data? Would you have preferred to see it represented in some other way?

2.5 Change settings

change the color of the heatmap was there a color you liked more? And which is clearer? adjust playback speed was it reasonable?

3 Question time

Which part was the hardest? Which part of the website was your favorite and why? Is there any feature that you would have liked that didn't seem to exist yet? If you had seen them on Naturhistoriska, what would have made you think it was more fun to use?

4 Pause for 1 minute

(Where they are not thinking about the website because we are asking questions or something)

5 Repeat all steps and time again (tests memorability)

(Does not need to ask the questions again but can say in advance that they are welcome to tell us if they think of anything new the second time they test) Did it feel easier to do all the steps now?

E

Qualitative User Tests Results

The results from the qualitative user tests are divided into one section per task. For each task, the responses from each of the six users are displayed. Firstly, background information about the users is listed down below.

Test Users Background
User 1: Male, 22 years old
User 2: Male, 22 years old
User 3: Male, 23 years old
User 4: Female, 25 years old
User 5: Female, 22 years old
User 6: Female, 22 years old

To understand what the users refer to, an image of the application front end is included here in figure E.1.

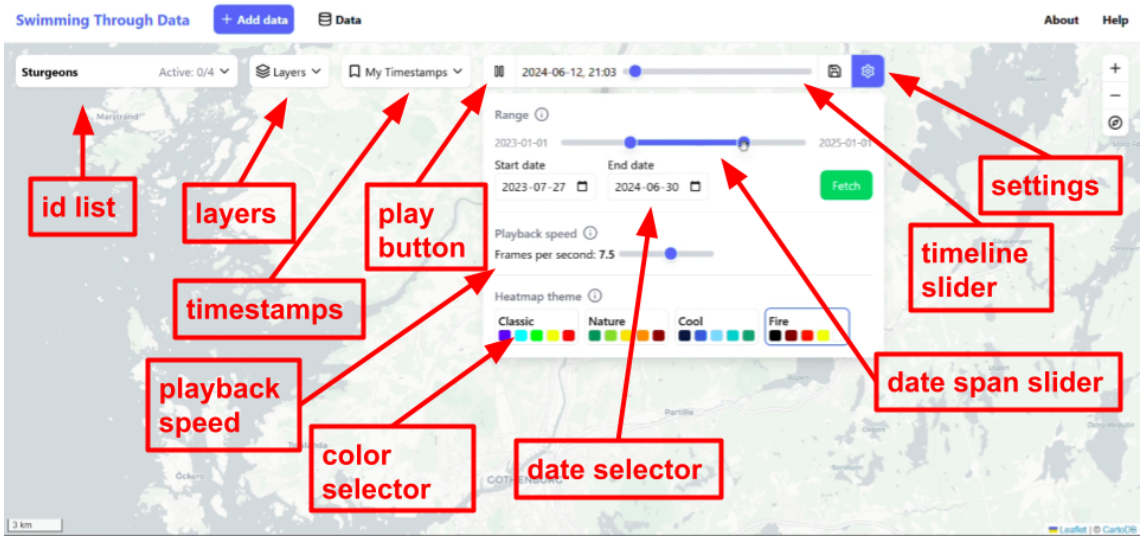


Figure E.1: Application front end with labels

Task 1: load data from excel files into program
User 1: When uploading the three Excel files, he wanted to select all three at the same time by using the keyboard button ctrl + selecting all of them. It would be

easier if the user gets redirected to the visualization page when the data has finished loading to avoid confusion about where to click.
User 2: All steps were clear and he understood them well.
User 3: He directly found the right page and found it easy to upload all files. Thought it was good that the user gets a pop-up message when the data has finished loading, although it could be made larger for better accessibility for people with visual impairment.
User 4: Found all steps easily. Did not read the pop-up on finished loading but understood what it meant by giving it a glimpse.
User 5: Tried to select multiple files at the same time using the ctrl button and selecting with the mouse. Otherwise, all steps were followed easily.
User 6: Had no negative feedback. Everything went as planned.

Task 2: select a time span to visualize
User 1: Firstly, they tried to adjust the animation time slider, but when nothing happened, he quickly clicked on the settings icon and found the date selector easy to use. He used tab to select month-day-year in the dates.
User 2: Tried to select a date in the time stamp menu, but when it did not work, he was quick to find the settings button and select a time span. Thought that it was unclear what the time stamp menu was.
User 3: Also tried to check the time stamp menu first, then tried the timeline selector and play button and finally found the settings button where he could easily select the date span. He reacted that the dates are defined in an American standard with month-day-year format.
User 4: The user tried the following interactions before finding the settings button: layers -> timestamps -> writing the date in timestamp -> timeline -> help button -> settings button. After finding the settings it was easy to select dates. She used the graphical selector and thought it was unusual to use the American date format. Her thought after finishing the task was that it would be more clear if the UI that is used to select part of the data and the UI that is used to visualize the data are separated and not hidden away in the same settings menu.
User 5: The following interaction before finding the settings menu was the following: data tab -> layers -> id list -> clicking play button -> tries the timeline bar -> checks timestamp list -> help button -> settings button. Using the graphical date selector, she found it easy to select the correct dates. Thought it should be named to something related to data to be found more easily.
User 6: Interactions here were: data -> time stamps -> layers -> id list -> settings. She was the first to use the slider to select the date. Would want to be able to write the date using numbers and a Swedish format.

Task 3: Interpreting the Heatmap
User 1: He firstly thought that it represented how many fish were in an area, but when we told him that it is only one selected fish he understood that it visualizes a fish that is migrating into the river from the ocean. He also understood that the color intensity represents how certain you are that a fish is in a certain location.

User 2: The user understood that the heatmap moves depending on where the fish has been. He thought that the color displays how often a fish has been at a certain sensor.
User 3: He thought the heatmap displays for how long a fish has been in an area and how far it has swum.
User 4: Thought that the heatmap displays where the fish has swum and the color intensity the probability that it is in an area.
User 5: Said that the heatmap displays where the fish has been. The fading "tail" she thought was a buffering effect at first glance, but after more thorough examination thought that it displays a normal distribution of the fish location.
User 6: Firstly thought that the heatmap displays where all fish have been, but after we told her that it is one fish, she thought it displayed the probability of the fish location.

Task 4: Selecting Color of the Heatmap
User 1: Liked all colors
User 2: The color classic was most clear. The color nature and the others blended in with the background color.
User 3: The color classic was the color that you intuitively thought was the most suited color. It was good because of the high contrast ratio. Also, the color cool was good, but he did not like fire (although nice name) and nature.
User 4: Classic was clearer and cool could work. The others were not good because of the lower contrast ratio.
User 5: She thought that the color fire was most clear as it is a single color scale where the brightness ratio is scaled.
User 6: Cool and classic was best, nature could work and fire was not clear.

Task 5: Define satisfaction- and learnability score
User 1: Satisfaction 4/5, Learnability 4/5
User 2: Satisfaction 4/5, Learnability 4/5
User 3: Satisfaction 4/5, Learnability 4/5
User 4: Satisfaction 5/5, Learnability 4/5
User 5: Satisfaction 4/5, Learnability 2/5
User 6: Satisfaction 4/5, Learnability 4/5

Bibliography

[1] Chandrashekar Althati, Manish Tomar, and Lavanya Shanmugam. "Enhancing Data Integration and Management: The Role of AI and Machine Learning in Modern Data Platforms". In: *Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023* 2 (Feb. 2024), pp. 220–232. DOI: 10.60087/jaigs.v2i1.154.

[2] SLU Artdatabanken. *Artfakta: atlantstör (Acipenser oxyrinchus)*. URL: https://artfakta.se/taxa/100002 (visited on 04/29/2025).

[3] Jake Brereton. *The Triple Diamond Design Process: A Comprehensive Guide*. Mar. 22, 2024. URL: https://www.launchnotes.com/blog/the-triple-diamond-design-process-a-comprehensive-guide (visited on 03/25/2025).

[4] Elliot J Brown et al. "Conflicts in the coastal zone: human impacts on commercially important fish species utilizing coastal habitat". In: *ICES Journal of Marine Science* 75.4 (Jan. 2018), pp. 1203–1213. ISSN: 1054-3139. DOI: 10.1093/icesjms/fsx237. eprint: https://academic.oup.com/icesjms/article-pdf/75/4/1203/31236889/fsx237.pdf. URL: https://doi.org/10.1093/icesjms/fsx237.

[5] Baltic Marine Environment Protection Commission. *Protection and Recovery of the Baltic Sturgeon*. 2019. URL: https://helcom.fi/wp-content/uploads/2020/06/HELCOM-Sturgeon-Action-Plan-2019-2029.pdf (visited on 05/19/2025).

[6] Wikipedia contributors. *Heat map — Wikipedia, The Free Encyclopedia*. 2025. URL: https://en.wikipedia.org/wiki/Heat_map (visited on 05/05/2025).

[7] Pete Koomen Dan Siroker. *A / B Testing: The Most Powerful Way to Turn Clicks Into Customers*. July 27, 2015. URL: https://books.google.se/books?hl=en&lr=&id=9xUWCgAAQBAJ&oi=fnd&pg=PA1&dq=performing+a/b+testing&ots=GD1Kay8B-7&sig=AzFmA-SWzS_Q2yJUtt0lfJcDTd8&redir_esc=y#v=onepage&q=performing%20a%2Fb%20testing&f=false (visited on 03/29/2025).

[8] Department of Census and Statistics. *Computer Literacy Statistics 2022: Annual Bulletin*. 2022. URL: https://www.statistics.gov.lk/Resource/en/ComputerLiteracy/Bulletins/AnnualBuletinComputerLiteracy-2022.pdf (visited on 05/19/2025).

[9] FastAPI. *FastAPI framework, high performance, easy to learn, fast to code, ready for production*. 2025. URL: https://fastapi.tiangolo.com/ (visited on 05/16/2025).

[10] Roy Thomas Fielding and Richard N. Taylor. "Architectural styles and the design of network-based software architectures". AAI9980887. PhD thesis. University of California, Irvine, 2000. ISBN: 0599871180.

[11] Python Software Foundation. *Python is a programming language that lets you work quickly and integrate systems more effectively.* 2025. URL: https://www.python.org/ (visited on 05/16/2025).

[12] Matthew H. Futia et al. "Modeling regional occupancy of fishes using acoustic telemetry: a model comparison framework applied to lake trout". In: *Animal Biotelemetry* 12.1 (Sept. 2024), p. 25. ISSN: 2050-3385. DOI: 10.1186/s40317-024-00380-3. URL: https://doi.org/10.1186/s40317-024-00380-3.

[13] GeeksforGeeks. *React Introduction.* 2025. URL: https://www.geeksforgeeks.org/reactjs-introduction/ (visited on 05/05/2025).

[14] Jörn Gessner. *Reintroduction of the European Sturgeon to Germany.* 2024. URL: https://www.igb-berlin.de/en/sturgeon (visited on 05/19/2025).

[15] Cameron Hashemi-Pour and Nick Barney. *What is named entity recognition (NER)?* 2024. URL: https://www.techtarget.com/whatis/definition/named-entity-recognition-NER (visited on 05/19/2025).

[16] Interaction Design Foundation - IxDF. *What is Heuristic Evaluation (HE)?* URL: https://www.interaction-design.org/literature/topics/heuristic-evaluation (visited on 04/29/2025).

[17] Kirill Karakhainko. *Choosing the Best Tech Stack for Web App Development.* 2023. URL: https://techwings.com/blog/choosing-the-best-tech-stack-for-web-app-development (visited on 05/05/2025).

[18] Ryszard Kolman, Andrzej Kapusta, and Arkadiusz Duda. "Re-establishing the Atlantic Sturgeon (Acipenser oxyrinchus oxyrinchus Mitchill) in Poland". In: Springer-Verlag Berlin Heidelberg, July 2011, pp. 573–581. ISBN: 978-3-642-20610-8. DOI: 10.1007/978-3-642-20611-5_43.

[19] Janet Kuhn. *Decrypting the MoSCoW Analysis.* URL: https://www.itsmsolutions.com/newsletters/DITYvol5iss44.pdf (visited on 04/28/2025).

[20] Leaflet. *Leaflet - an open-source JavaScript library for interactive maps.* URL: https://leafletjs.com/ (visited on 05/05/2025).

[21] Leaflet. *Leaflet.heat: A tiny, simple and fast heatmap plugin for Leaflet.* 2025. URL: https://github.com/Leaflet/Leaflet.heat (visited on 05/05/2025).

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[23] Heeseok Lee. "Justifying database normalization: a cost/benefit model". In: *Information Processing & Management* 31.1 (1995), pp. 59–67. ISSN: 0306-4573. DOI: https://doi.org/10.1016/0306-4573(95)80006-F. URL: https://www.sciencedirect.com/science/article/pii/030645739580006F.

[24] Livelox. *Livelox: Orienteering Analysis Tool.* 2025. URL: https://www.livelox.com/ (visited on 05/05/2025).

[25] Eva Lundgren. *Atlantstör från Göta älv återfunnen i Norge.* URL: https://www.gu.se/nyheter/atlantstor-fran-gota-alv-aterfunnen-i-norge (visited on 05/19/2025).

[26] University of Massachusetts Amherst. *How to Conduct Usability Testing for UI/UX Design*. Jan. 15, 2024. URL: https://bootcamp.umass.edu/blog/ui-ux/how-to-conduct-usability-testing-for-ui-ux-design (visited on 03/25/2025).

[27] Microsoft. *TypeScript is JavaScript with syntax for types.* 2025. URL: https://www.typescriptlang.org/ (visited on 05/16/2025).

[28] Rohit Nishant, Dirk Schneckenberg, and MN Ravishankar. "The formal rationality of artificial intelligence-based algorithms and the problem of bias". In: *Journal of Information Technology* 39.1 (2024), pp. 19–40. DOI: 10.1177/02683962231176842. eprint: https://doi.org/10.1177/02683962231176842. URL: https://doi.org/10.1177/02683962231176842.

[29] Jesse A. Nunez, Dashi I. Singham, and Michael P. Atkinson and. "A particle filter approach to estimating target location using Brownian bridges". In: *Journal of the Operational Research Society* 71.4 (2020), pp. 589–605. DOI: 10.1080/01605682.2019.1570806. eprint: https://doi.org/10.1080/01605682.2019.1570806. URL: https://doi.org/10.1080/01605682.2019.1570806.

[30] OpenStreetMap. *Data Derived from OpenStreetMap for Download.* URL: https://osmdata.openstreetmap.de/data/water-polygons.html (visited on 05/02/2025).

[31] OpenStreetMap. *Overpass API.* URL: https://wiki.openstreetmap.org/wiki/Overpass_API (visited on 05/02/2025).

[32] Aiswarya Raj et al. "Modelling Data Pipelines". In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020, pp. 13–20. DOI: 10.1109/SEAA51224.2020.00014.

[33] Drew Roselli, Jeanna Matthews, and Nisha Talagala. "Managing Bias in AI". In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. San Francisco, USA: Association for Computing Machinery, 2019, pp. 539–544. ISBN: 9781450366755. DOI: 10.1145/3308560.3317590. URL: https://doi.org/10.1145/3308560.3317590.

[34] SMHI. *Radar med blixt - SMHI.* 2025. URL: https://www.smhi.se/vader/radar-och-satellit/radar-med-blixt/q/norden/plats/Stockholm/2673730 (visited on 05/05/2025).

[35] SMHI. *Sveriges största vattendrag.* URL: https://www.smhi.se/kunskapsbanken/hydrologi/sveriges-vattendrag/sveriges-storsta-vattendrag (visited on 03/05/2025).

[36] Meta Open Source. *The library for web and native user interfaces.* 2025. URL: https://react.dev/ (visited on 05/16/2025).

[37] Ramya Srinivasan and Ajay Chander. "Biases in AI Systems: A survey for practitioners". In: *Queue* 19.2 (May 2021), pp. 45–64. ISSN: 1542-7730. DOI: 10.1145/3466132.3466134. URL: https://doi.org/10.1145/3466132.3466134.

[38] Strava. *Strava Global Heatmap.* 2025. URL: https://www.strava.com/maps/global-heatmap?sport=All&style=satellite&terrain=false&labels=true&poi=true&cPhotos=true&gColor=blue&gOpacity=100#9/37.7749/-122.4194 (visited on 05/05/2025).

[39] Emma Strubell, Ananya Ganesh, and Andrew McCallum. “Energy and Policy Considerations for Deep Learning in NLP”. In: *CoRR* abs/1906.02243 (2019). arXiv: 1906.02243. URL: http://arxiv.org/abs/1906.02243.

[40] Gunnar Taraldsen, Tor Reinen, and Tone Berg. “The underwater GPS problem”. In: *OCEANS, 2011 IEEE - Spain*. July 2011, pp. 1–8. DOI: 10.1109/Oceans-Spain.2011.6003649.

[41] Havs- och Vattenmyndigheten. *Atlantstör*. URL: https://www.havochvatten.se/arter-och-livsmiljoer/arter-och-naturtyper/atlantstor.html (visited on 04/29/2025).

[42] Göta älvs vattenvårdsförbund. *Fakta om Göta älv: En beskrivning av Göta älv och dess avrinningsområde nedströms Vänern 2015*. URL: https://www.gotaalvvvf.org/download/18.2f0ad835166c596881356a83/1540998119692/fakta_om_gota_alv_webb.pdf (visited on 03/05/2025).

[43] Wikipedia contributors. *Date and time notation in Sweden — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/w/index.php?title=Date_and_time_notation_in_Sweden&oldid=1257523326 (visited on 05/19/2025).

[44] Wikipedia contributors. *Date and time notation in the United States — Wikipedia, The Free Encyclopedia*. 2025. URL: https://en.wikipedia.org/w/index.php?title=Date_and_time_notation_in_the_United_States&oldid=1290181273 (visited on 05/19/2025).

[45] Wikipedia contributors. *Shapefile — Wikipedia, The Free Encyclopedia*. 2025. URL: https://en.wikipedia.org/w/index.php?title=Shapefile&oldid=1283574959 (visited on 05/02/2025).