Design Report

Development of a Web-Based Application for Water Pipeline Inspection and Marker Management

Faculty of Electrical Engineering, Mathematics
and Computer Science

Supervised by
Dr. L. Ferreira Pires

Authors
Tintin Wongthanaporn (s2712911)
Jason Hsu (s2708442)
Jump Srinualnad (s2690837)
Yixuan Zhuang (s2200848)
Thijs Beumer (s2761831)

Date - April 19th, 2024

Table of Contents

1 Introduction

In this section we will give an introduction to our project and how it was approached, as well as giving a structural outline of this report.

1.1 Problem statement & motivation

Rosenxt, a spinoff of Rosen, is a company that specializes in the inspection of water supply pipelines. They have developed a solution to verify the integrity of water supply lines that can help determine weak points within a line before they become a problem, allowing for preventive and corrective maintenance long before critical failures. Their devices collect information such as video and ultrasonic among others to find and detect these possible points of failure.

Before such an inspection device can be planted into a pipeline, a good knowledge of the route of the line is necessary. This knowledge can then be used to add physical markers on the pipeline which can detect when the inspection device passes through the pipeline underneath and get some data from the device. The handling of this data is being done across a large variety of different mediums making it hard to keep track. With our application, we aim to give a centralized system for storing and manipulating this data providing easier access to this information.

1.2 Objective

To solve the problem at hand, some required functionalities have been specified. The most important functionality is being able to add, edit, and delete pipeline and marker information to the system, this allows for the centralized overview of all data about these inspection runs. We aim to create a user-friendly interface to import this data as well as allow other software to connect to our application in order to automate such a process.

1.3 Approach

The first phase of the project is the design phase, in this phase we will gather all the requirements by discussion with our client and create a well-structured overview of the functionalities that should be implemented. Next the development phase starts, where we will use the previously specified design plans to realize the functionalities into a working system. During this phase, the system will also be tested by utilizing both automated and manual testing approaches to ensure that we create a robust product that will not break in real-world scenarios. In the end, there will be a finalization phase where we properly document all implemented functionalities, present our application, and create a manual on how to use the system effectively.

1.4 Report structure

This report will start off with a description of the project management methods we have used during the project. Subsequently, an overview of the specified requirements will be given as agreed upon with our

client. In the next section, a detailed overview of all implemented functionalities as well as our choices of implementation such as programming languages and code structure is provided. After this overview will be a section about how we tested the system to make sure that our application is robust and to finalize our report, we will reflect on our work as a group.

2 Project Management

In this chapter, the used methods for managing the different aspects of the project are discussed. To properly manage everything from human resources to our codebase, we decided to use some well-known and proven to be effective management methods.

2.1 Agile Project Management

To manage the flow of the entire project we decided on using the Agile SCRUM method (Srivastava et al., 2017). Work is completed with an iterative approach, the project is broken down into small manageable, and tangible sections. This allows for a customer-centric approach where after every period of work on the project, there is a new functionality that can be shown to the customer. The customer can, with this approach, better follow and understand progress on the project and set realistic expectations. Another important aspect of the SCRUM methodology is the flexibility it creates, by sectioning the work into manageable pieces, it is easier to have possible swaps in product requirements over the course of the development.

2.2 Team Management Tool

Insight into work and progress is managed by the use of JIRA software (Atlassian, 2023). The software offers a way to describe our defined user stories and track progress per story by assigning statuses such as “open”, “in progress”, “testing” and “done”, and also dividing work into sprints. This gives better insight into the work that still needs to be done and also the progress of parts of the system that are being worked on at any time. An overview of the work that has been completed in every iteration (sprint) of the project can be found in Appendix E.

2.3 DevOps Platform

We use GitLab as our version control system for the system development. We have a strict Git Flow process and CI/CD framework and the management of Git is linked with the management of the Jira board, which means that we as developers can easily manage the process of the project development.

We have multiple branches in the development process, which include ‘master’ and ‘develop’. The master branch is the branch that always contains a working version of the application. The develop branch contains newly added features that have already been tested and approved. The new features of the system are assigned to developers via the Jira board. The developer starts developing the features by creating a new branch from the develop branch and naming it following a strict naming rule so the team can know which branch is for which feature.

After developing the new features, the developer needs to create a merge request and at least one of the other team members needs to test and approve it before it can be merged into the develop branch.

2.4 Communication with the client

A sprint review meeting is scheduled every two weeks on Thursday. In the sprint review meeting, the client checks the current design choices to satisfy their requirements and gives feedback for future development. The meeting records can be found in Appendix C.

We also communicate with the client by email. However, they are usually too busy to answer the emails which causes some development delays.

2.5 Communication with the supervisor

We have a weekly meeting with the supervisor every Friday. The purpose of the meeting is to keep the supervisor updated on what is going on with the project. Moreover, we can have feedback from the supervisor to ensure that the project is on the right track. The meeting records can be found in Appendix D.

3 Requirement Analysis

This section contains the requirement analysis, where we compile and formalize requirements from meetings and project descriptions from clients, which will help us split the work, time estimation, and reduce ambiguities.

To ensure that the requirements identified in this analysis are clear, actionable, and measurable, we will be following the SMART guidelines and the MoSCow prioritization framework.

SMART Guidelines:
Each requirement will be evaluated against the SMART criteria to ensure that it is Specific, Measurable, Achievable, Relevant, and Time-bound, where the Time-bound property will not be included in the requirement but rather will be divided into sprints. We will be utilizing the Jira board to include all these criteria allowing clear expectations and tracking progress.

MoSCoW Prioritization:
In addition to the SMART guidelines, we will prioritize requirements using the MoSCoW framework, which helps categorize requirements into 4 different priority levels:

- **Must Have:** Requirements are essential for the minimum viable product and will be the focus point of the first 2 sprints.
- **Should Have:** Requirements that are important but are not needed for the minimum viable product. These requirements may be deferred to a later sprint if necessary, but efforts will be made to include them in the final product.
- **Could Have**: Requirements that are nice to have but are not essential to the project's success. These requirements may be considered in the last few sprints if there are time and resource permits but can be deprioritized if necessary.
- **Won't Have:** Requirements that will not be included in the current project but are some of the requirements that have been deeply considered.

3.1 Stakeholder

Key stakeholders:

1.) Rosenxt employees: they will directly use the application to analyze the inspection data. Most information about the project will be provided by them.
2.) Rosenxt's client: the system will contain information about the client including the route and condition of the pipeline. However, they will not have access to the application.

3.2 Stakeholder Requirements

1. **As a user, I want to log in and out of the system**
   This requirement ensures that the user can only access the web application by logging in with their credentials and logging out of the system when they are done.

2. **As a user, I want to navigate the map by zooming and moving around with my cursor**
   This requirement allows the user to navigate through the map interface freely by dragging the mouse's cursor and zooming in and out using the mouse scroll wheel. This makes exploring the map flexible and easy.

3. **As a user, I want to view markers/inspection points along the pipeline**
   This requirement enables us to view markers along the pipeline on the map. Users can name the markers and leave some useful information on the marker's detail so it will be useful when looking back to why the marker was placed.

4. **As a user, I want to view the route of a pipeline on a map**
   This requirement enables users to visualize the geographical route of the pipeline on the mapping interface. It is useful for users to view the pipeline digitally on the map because it provides essential spatial information.

5. **As a user, I want to be able to know the distance between two markers**
   This requirement allows the user to determine and optimize a better location to place the marker. Since sometimes markers that are too close to each other would not give any valuable information about the pipeline. The distance between two markers also helps with planning or analysis purposes.

6. **As a user, I want to be able to add onsite survey data to markers**
   This requirement allows the user to note extra information about the location on each marker. This will contain information such as differential GPS, depth, and notes. It is important because it provides relevant information associated with specific locations.

7. **As a user, I want to have the option to start a navigation route to a given marker using a 3rd party navigation system**
   This requirement will provide users with a button on the sidebar to a link to Google Maps, this link will automatically fill in latitude and longitude as a destination so the user can navigate there.

8. **As an administrator, I want to manipulate (create, edit, delete) pipeline routes on the map**
   This requirement grants the administrator the ability to perform various actions related to the pipeline routes, including creating new routes, modifying an existing route, and deleting any routes.

9. **As an administrator, I want to manipulate (create, edit, delete) markers along the pipeline**
   This requirement grants the administrator the ability to manage markers or inspection points that are placed along the pipeline route. Administrators can create new markers, edit existing markers' information, and delete unnecessary markers as needed.

10. **As an administrator, I want to administer (create, edit, delete) user accounts**

This requirement enables the administrator to administer user accounts within the system including creating new user accounts, modifying existing user profiles, and deleting user accounts when necessary. This ensures effective user management and system security.

3.3 System Requirements

1. **The system must have a login and logout functionality**
   The user must be able to securely log in to the system using their credentials and log out when they are done.

2. **The system must include a mapping interface the user can navigate by zooming and moving around**
   The system should be able to provide users with a user-friendly mapping interface. Users can interact with the map by zooming in and out, panning, and moving around the map. This makes navigating through the map easy.

3. **The system must have a mechanism to manipulate (create, edit, delete) markers**
   The system should allow users to create markers on a specific location on the map, edit any existing markers, and delete unnecessary markers. This allows the user to manage and update relevant information on the markers.

4. **The system must have a mechanism to manipulate (create, edit, delete) users**
   The system should grant the ability for the administrator to manage users' accounts, including creating a new account, editing an existing profile, and deleting any accounts when necessary.

5. **The system must calculate the distance between any two markers along the pipeline**
   The system should be able to calculate the distance between any two markers so that this can be displayed for the user. This provides valuable information and gives the user a better understanding and idea of where the optimal distance between two markers is.

6. **The system should have a mechanism to manipulate (create, edit, delete) pipeline routes**
   The system should allow administrators to manage pipeline routes, including creating new routes, editing the existing routes, and deleting any unnecessary routes.

7. **The system should have an integration with 3rd party navigation systems to allow for navigation to a specific marker**
   The system should display a button when a marker is selected, redirecting users to a third-party navigation system for the best route to that marker.

8. **The system should show to which client the route is linked**
   The system should indicate which client the route is linked to in the sidebar area when a route is selected.

9. **The system should display survey information related to any specific marker**

The system should be able to display survey information details in the sidebar area when a marker is selected.

10. **The system <u>should</u> have a mechanism to add onsite survey information to a marker**
    The system should allow users to add survey information directly through the sidebar when a marker is selected and save the information in the database correctly and safely.

11. **The system <u>should</u> be able to export the pipeline and markers information in a CSV file**
    The system should have a button that allows users to export the selected pipeline and marker information in a CSV formatted file.

12. **The system <u>could</u> implement an automated system for suggesting optimal marker placements**
    The system could have a button that allows users to automatically generate markers that have a fixed distance between them on a selected route.

13. **The system <u>could</u> contain API to import pipeline trajectory points in a CSV file**
    The system could have a button that allows users to import the pipeline trajectory points that are in CSV file format.

14. **The system <u>won't</u> have an integration with already existing Rosenxt systems**
    The system will be a stand-alone proof of concept application. There won't be any connection to already existing data streams or processes within the Rosenxt company.

15. **The system <u>won't</u> contain offline mode functionalities**
    The system will only be functional when the user has a connection to the internet.

3.4 Quality Requirements

- **The system <u>must</u> respond to user interactions within 2 seconds to ensure a smooth and responsive user experience.**
  The system must respond to the user interactions within 2 seconds to ensure a smooth and responsive user experience. This requirement makes sure that the user will be satisfied with the system by ensuring that the system reacts promptly to the user input. A quick response time generally contributes to a better user experience.

- **The system <u>must</u> implement encryption protocols, such as TLS/SSL, to protect sensitive data transmitted over the network.**
  The system must implement encryption protocols to protect sensitive data transmitted over the network. This requirement focuses on secure data transmission, by encrypting data before being sent over the network, making it unreadable to unauthorized users to reduce risk.

- **The system <u>must</u> be compatible with external systems and platforms, allowing for seamless integration and data exchange.**

The system must be compatible with external systems and platforms. Allowing integration and data exchange emphasizes interoperability by ensuring that the system can communicate with other systems and share data without compatibility issues.

- **The system must validate user inputs and prevent data entry errors to maintain data accuracy.**
  The system must validate user inputs and prevent data entry errors. This requirement makes sure that only valid and accurate information is stored in the system. Input validation helps prevent data entry errors such as typos or incorrect formats, reducing the risk of data corruption.

- **The system should support deployment in cloud environments, such as AWS or Azure, to facilitate scalability and flexibility.**
  The system should support development in cloud environments. This requirement emphasizes scalability and cost-effectiveness by leveraging cloud infrastructure and services. Cloud enables the system to scale resources to accommodate changing workloads and optimize performance.

- **The system should be designed with testability in mind, with modular components that can be easily isolated and tested in isolation.**
  The system should be designed with testability in mind. The requirement ensures software equality and maintainability by facilitating testing throughout the development lifecycle. This makes it easy to identify and fix bugs, therefore enhancing the readability and ensuring robustness.

3.5 Security Requirements

- **The system must handle passwords in a secure way.**
  The system must have strong password policies. This requirement aims to enhance security by ensuring that user passwords meet certain criteria that make them difficult to crack. By having strong password rules, the system reduces the risk of unauthorized access due to weak or easily guessable passwords. Implementing a password expiration period to require regular password updates makes the system more secure.

- **The system must have a role-based authorization mechanism.**
  The system must have ROle-based access control. This requirement demands assigning roles to users based on their responsibilities and authorizing access to the system. RBAC ensures that the users only have access to the information and the functionalities of their roles. This minimizes the risk of unauthorized access to sensitive information.

- **The system must provide a mechanism for users to manually revoke or terminate their sessions.**
  This requirement empowers the users to manage their session security by allowing them to revoke active sessions that they no longer need. By enabling users to terminate their sessions remotely, the system reduces the risk of unauthorized access from devices or locations that users no longer have access to.

- **The system must implement robust session management mechanisms to ensure the secure initiation, maintenance, and termination of user sessions.**
  This requirement encompasses various security measures, such as secure initiation protocols and session timeout policies. By implementing secure session management, the system minimizes the risk of session hijacking and unauthorized access thereby protecting sensitive information and protecting user privacy.

- **The system won't contain session persistence across multiple devices for improved user experience**.
  The decision not to include session persistence across multiple devices in the system is that it is unnecessary for the intended functionality and the objective of the system. While session persistence can enhance user experience, it introduces complexity in implementation. Users will primarily interact with the system through a single device so we will omit this functionality.

- **The system won't have Multi-factor authentication (MFA) to add an extra layer of security.**
  This requirement states that the system will not employ multi-factor authentication, which typically involves requiring users to provide multiple forms of authentication such as passwords followed by a security token. Although authentication is a part of this project, security is not the main concern. We will implement basic security features but leave extended security such as MFA as a possible future improvement to allow focus on other requirements.

4 Detailed Design

This section reflects on the design choices that the team has made during the development process. First, an overview of the system is provided. Then, some diagrams are provided to more intuitively display the structure of the system. Finally, the important design choices are given and discussed in detail.

4.1 Graphical User Interface Overview

This section gives an overview of different pages of the application. In Appendix A, we provide a user manual where all the functionalities of the application are shown and explained.

4.1.1 Login page

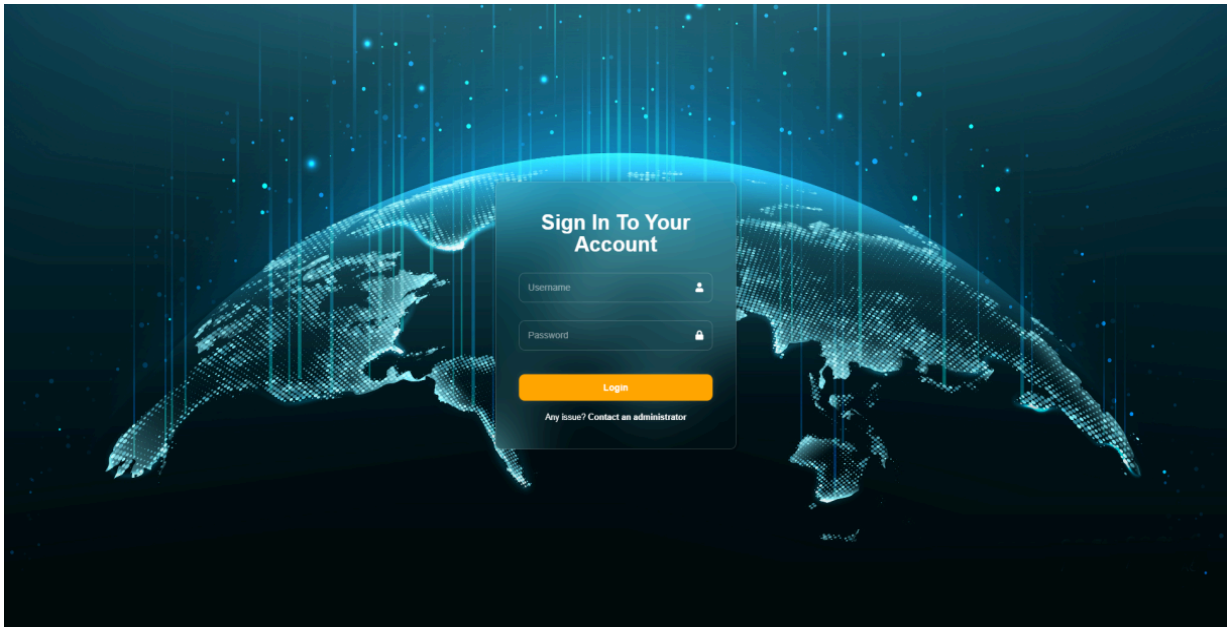


Figure 4.1: Login Page

The Login page is the first page users see when they launch the application. Users can log in to the application using their email address and password on this page. In case the user forgets their credentials or has difficulty logging in, see No “forget password” choice in the login screen section.

4.1.2 Home Page

The Home page is the main page that contains the core functionalities of the application. Users can see a map with pipelines and markers. There are two modes named Add Marker mode and Show Sidebar mode which allow users to add markers to the pipelines and to inspect information of pipelines and markers respectively. Users can switch modes easily by clicking buttons on the top of the interface. Besides the switching mode buttons, there is a Create Pipeline button which allows users to create new pipelines by importing CSV files or manually entering the coordinates of points on the pipeline. Users can also select different pipelines on the top right drop-down menu. On the left bottom corner, users can change different

display modes of the map and turn on the distance between the markers' display. Last but not least, users can let the system automatically add markers to the selected pipeline with a fixed distance between markers.

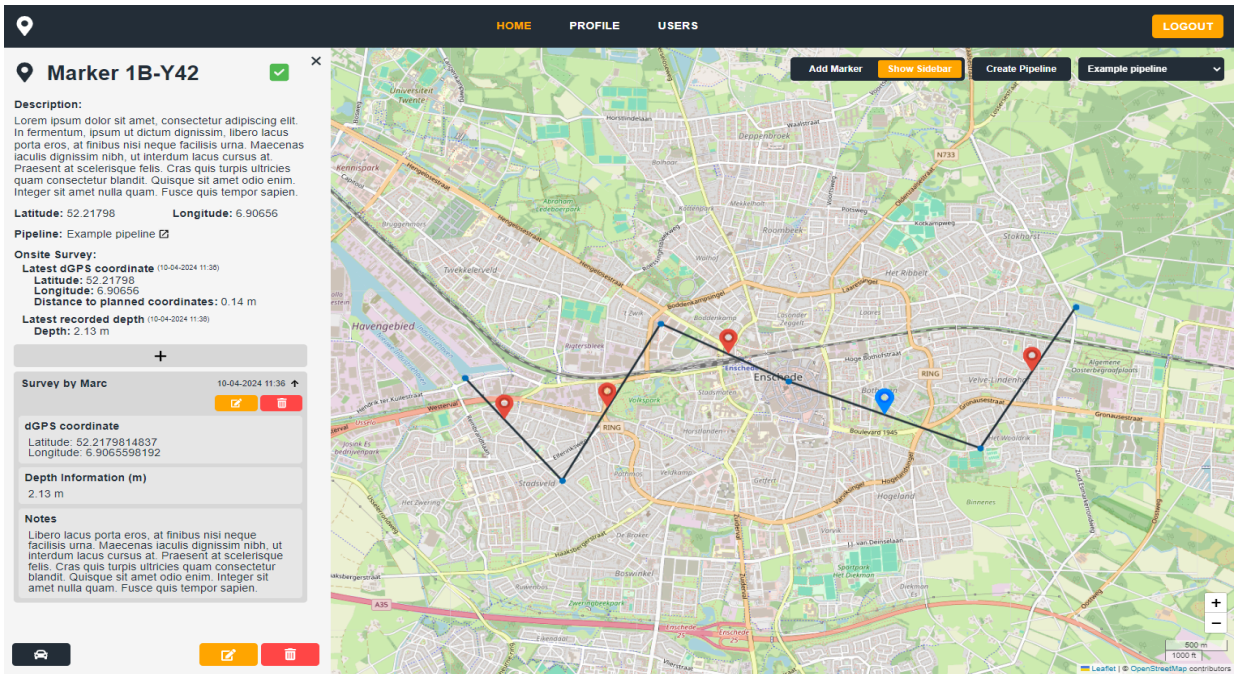


Figure 4.2: Home Page

4.1.3 Profile Page

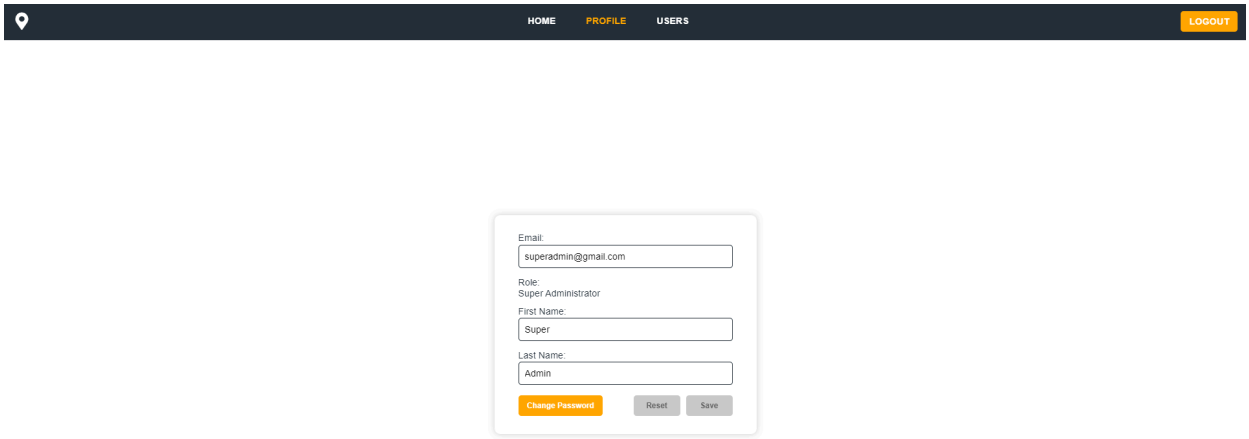


Figure 4.3: Profile Page

The Profile page contains all the information of the current user, including email address, role, first name, and last name. The user can edit their email address, name, and password on this page.

4.1.4 Users Page

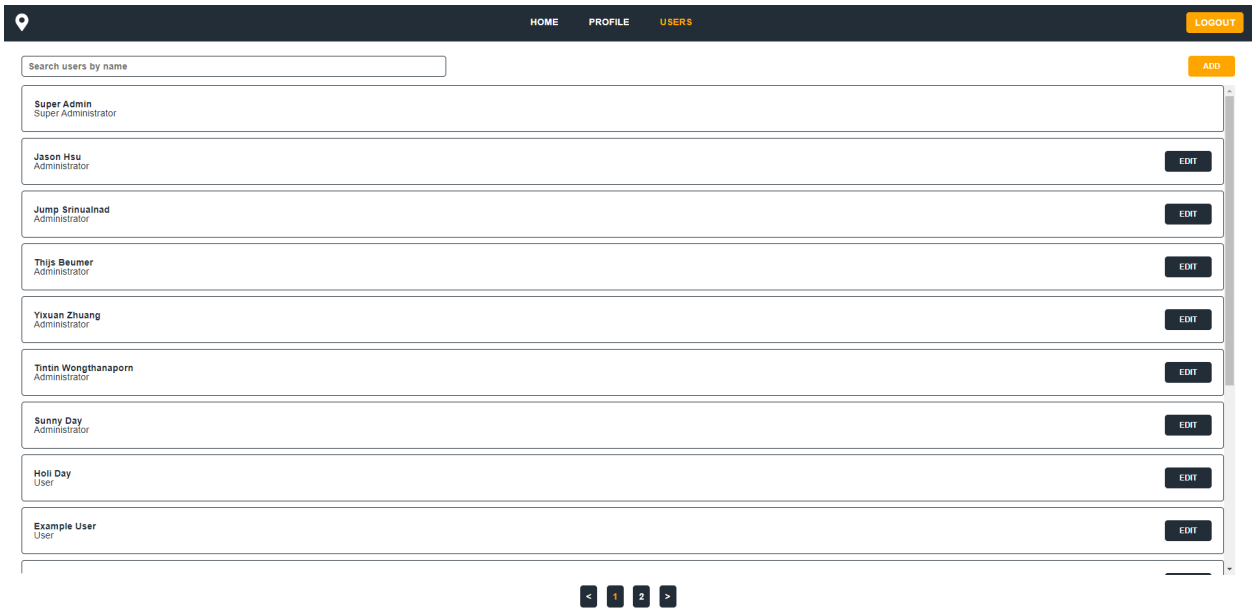


Figure 4.4: Users Page

The Users page is only visible to users who have Super Administrator and Administrator roles. The super admin can add an admin user and a normal user to the system while the admin user can only add a normal user to the system. The super admin users can edit the details of admin users and normal users and have the privilege to delete them. The admin users can only edit the details of the normal users and delete them.

4.2 System Description

This section provides an overview of the components and functionality of the system.

4.2.1 Architecture Overview

The applications follow the client-server architecture paradigm (Lile, 1993), a fundamental design approach where distinct components, namely the client and server, are developed and deployed independently. This architectural setup enables flexibility and scalability, allowing updates or modifications to be implemented on one side without affecting the other.

The communication between the client and server takes place securely through HTTP protocols, laying the foundation for efficient data exchange and interaction. However, this initial setup is temporary and subject to transition. Upon deployment to the live environment, the system undergoes a crucial enhancement by switching to HTTPS protocols.

In addition to secure communication through HTTPS protocols, the application employs session cookies to facilitate user authentication and maintain user sessions. Upon successful login, a session cookie is generated and stored on the client side. This cookie serves as a unique identifier for the authenticated user, enabling the server to recognize and track their interactions throughout the session. Through the combined utilization of HTTPS protocols and session cookies, the application will meet the current industry security standards.

4.2.2 Components and functionalities

In Figure 4.5, these 6 classes represent part of the data stored in the database. For a complete view of the database structure, please refer to the Database structure section.

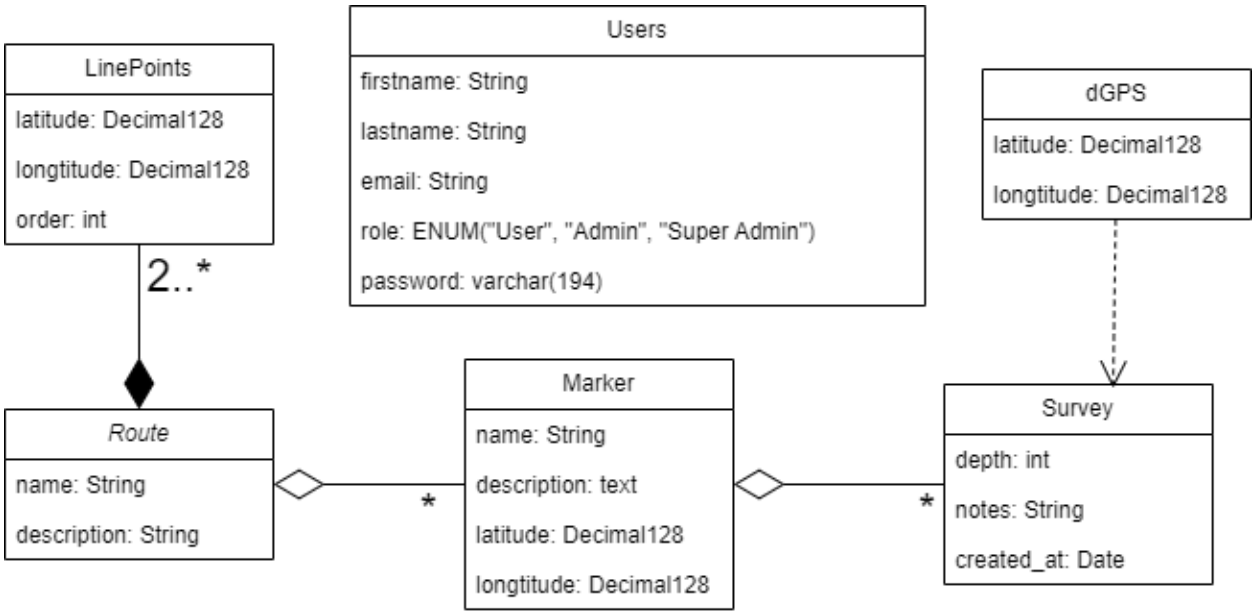


Figure 4.5: Main class diagram for the system

Class “Route” is the main core of the application which represents the route that needs to be inspected. A route has a name and description to help the employee differentiate each pipeline.

The class “LinePoints” represents a turning point in the pipelines. The latitude and longitude are the coordinates of where these turning points occurred and the order represents the direction of the pipelines.

The class “Marker” represents a point of inspection. Each pipeline route may have multiple markers and it is characterized by the latitude and the longitude, which are the coordinates of the marker. These markers' coordinates will always be a point on the pipeline.

The class “Survey” represents the data collected from each inspection. The system will always store when the survey is created. The survey can store depth information, notes, and Differential GPS coordinates which are represented as the class “dGPS”.

The class “User” represents the account that Rosen employees can use to access the information in the system. It contains information about employees including first name, last name, and email address. Moreover, the class “User” includes a role field that determines the level of permissions granted to the user within the system. Here is the description of each role:

- “Super Administrator” is reserved for a single user and created at the inception of the application. This account holds the highest level of access and is used to establish subsequent administrative roles.
- "Administrator" users possess similar privileges to "Super Administrator" but are unable to create the user of the role “Administrator”. Their role primarily entails administrative tasks within the system.
- "User" roles permit access to view all system resources but restrict interactions, except for surveys. Users with this role can create surveys but are unable to edit or delete them.

4.3 Design Choices

In this section, we will go through several major design choices that we made during the project. All of them are decided after clear consideration.

4.3.1 System look and feel

Our product serves as a proof of concept and is not intended for integration with existing systems. This freedom allowed us to design the system's appearance ourselves. Thus, throughout the project, our goal was to keep our application interface clean and simple to use and to ensure consistency by applying the same theme across all elements of the system interface.

4.3.2 No “forget password” choice in the login screen

Rather than including a "Forget Password" option on the login screen, we only ask users who forget their passwords to contact the administrator for assistance. There are two primary reasons for this decision. Firstly, the application is intended for use within a specific company, where the administrator is readily available to provide support. Secondly, given the project's scope and priorities, implementing a "Forget Password" feature was not deemed essential. This decision enabled us to focus on more useful features of the application.

4.3.3 Marker clustering

The third design choice is the inclusion of marker clustering. To fulfill the requirement for “optimal marker placement”, which will result in numerous markers along a single pipeline, we have decided to implement cluster markers. When zoomed out, marker clustering prevents all the markers from overlapping and even indicates how many markers are in a region. This feature helps us maintain the cleanliness of the application, which is one of our goals for the system's overall feel.

4.3.4 Non-draggable sidebar

During the implementation of the sidebar, we deliberated on whether to make the sidebar draggable or not. After experimenting with the draggable version, we concluded that it did not yield satisfactory results. In fact, if the sidebar is draggable, users might drag it to a position where it becomes inaccessible within the application, which we considered difficult to fix in a short amount of time. Thus, we have decided to maintain the sidebar at a fixed position, which we later found to be more effective for user navigation and interaction, particularly when the sidebar contains a significant amount of data.

4.3.5 Prediction of Marker Placement

In the add marker mode, users are presented with the expected locations of markers. However, we encountered a choice between two methods of display: showing the markers only when the mouse hovers over the line, or displaying them continuously. The first option offers the advantage of displaying markers only when needed, but it suffers from the small challenge of triggering the hover event on thin pipelines. Conversely, the second option provides constant visibility of the predicted marker placements. However, it comes with the drawback of resource inefficiency due to the continuous calculation of closest coordinates. Prioritizing user experience, we decided on the latter option, considering that browser lag is unlikely unless dealing with an excessive number of markers and lines, which is rare in practical scenarios.

4.3.6 Confirmation modal

The sixth design choice involves adding a confirmation popup to various functionalities, such as deleting users, pipelines, markers, or surveys, and suggesting markers. These functionalities were prone to accidental triggering initially, as users could simply click on a button to delete some objects or add multiple markers. However, if users inadvertently activate these functionalities, their actions cannot be reversed, and this will require additional time to rectify any mistakes made. To mitigate this issue, we decided to implement a confirmation popup before the user's request is finalized. This feature allows users to reconsider their actions before proceeding with either the deletion of some objects or the addition of suggested markers, thereby reducing the likelihood of unintended actions.

4.3.7 Two modes in the Home page

On the Home page of the application, we have separated some functionalities that have the same trigger method into two distinct modes: Add Marker mode and Show Sidebar mode. For example, without this separation, when a user clicks on a pipeline, it may be unclear whether they intend to add a marker or select the pipeline and open its sidebar. Thus, this separation ensures that users are not confused by a complex interface and can easily navigate between different tasks.

4.3.8 Toggle map style

The eighth design choice is the inclusion of a topographic map as an alternative map style within the application. By default, the application uses the basic map style provided by the Leaflet library. However, given that our project focuses on pipelines and onsite survey data collection, it would be nice if we could provide users with a map that shows clearer geographical features like terrain. Nonetheless, it is important

to note that the drawback of the topographic map is its longer loading time. Therefore, we have decided to offer users the functionality to switch between the default map and the topographic map. This allows users to choose the map style that best suits their needs and preferences.

4.3.9 Show distance switch

As the number of markers on a pipeline increases, distance labels may overlap, which can result in a cluttered appearance on the map. However, removing these labels entirely is not feasible as users may require distance information to determine marker placement. To address this, we have implemented a toggle switch that allows users to show or hide distance labels based on their preferences.

5 Implementation Choices

Considerations like maintainability and scalability are important when coming up with a good design. This section will focus on the choices of programming languages, frameworks, and libraries we made in the earlier stages of the project. Furthermore, we will explain the naming convention for URLs to ensure clarity and consistency within our web application. Lastly, we will discuss the type of database we used in the project and explain the database structure.

5.1 Used technologies

To create the system we had to make a selection for which technologies we would use. We opted to use NodeJS as our main run-time environment for both the back-end and the front-end. This allowed us to have the same environment for both sides of the application while still being able to develop them separately.

In the front end, the React library has been chosen to build interactive web interfaces. React is well-known for its component-based architecture that allows developers to reuse UI components. This feature assists us in managing and maintaining the codebase more efficiently, and it is the main reason that we selected this JavaScript library. Additionally, because several team members already have experience with the React library in other projects, we can easily help each other when any problem arises. Even if they cannot solve it immediately, the large React community, providing extensive documentation and third-party libraries, allows us to find a solution easily most of the time.

In addition to React, as the application must allow users to work on the map, we need a library for an interactive map; therefore, Leaflet is chosen. The Leaflet library provides a simple and lightweight solution for integrating maps into our web application. It also allows us to customize the map with various plugins such as “draw”, “marker-cluster”, and others, which helps us implement the map functionality more easily. The Leaflet library implements the interactive map interface in React by using the OpenStreetMap dataset. OpenStreetMap is a dataset with map imagery that is maintained by a community of contributors who make sure the map stays up to date.

In the back end, Express has been selected as the Node.js web framework that we will use to build RESTful APIs. It simplifies the process of defining routes and allows the use of middleware functions to handle tasks such as authentication and error handling. Since several team members have experience with this Node.js framework, we decided to use it from the start. With Express, we also use "Express Session", a library and middleware for Express.js, to simplify session management. It helps ensure that the user is logged in and authorized to access resources on the server for each request.

In addition to Express Session, we choose to use the "bcryptjs" library to address other security aspects of the application. "bcryptjs" securely hashes passwords using the bcrypt algorithm, which has the strength of automatic salting, so it can protect our application against threats such as rainbow table attacks and brute-force attacks.

5.2 Naming Convention of URLs

The naming convention for URLs applied in this project follows RESTful API conventions. Correct URLs must be sent to the server to perform operations on resources. The fundamental operations include Create, Read, Update, and Delete, and each of them has a convention defining the structure of URLs. For instance, creating a user involves a URL structure like "/api/users", while deleting a user utilizes a URL structure of "/api/users/{id}" with an id parameter specifying the user to be deleted. The choice to prefix every URL going to the server with "/api" serves to differentiate between client-side and server-side routes. It also allows other developers to immediately know that the route is intended for API functionality. The "/users" part of the URL depends on the resource the client side is requesting, but the overall structure of the URL looks similar and aligns with RESTful API conventions. The full list of URLs can be found in Appendix B.

5.3 Database

The database we chose is a NoSQL database called MongoDB. This database has a schema-less design allowing faster iteration and adaptation which aligns with our 2-week sprint duration. Moreover, MongoDB's official Node.js driver and various third-party libraries make the integration seamless.

5.3.1 Database structure

As depicted in Figure 5.1, the database contains a total of 7 schemas, categorized into 5 collections and 2 subdocuments, and is created according to Figure 4.5. This section provides a detailed explanation of the rationale behind the database's structural organization.

Firstly, **User** collection is the most intuitive to be defined. It contains id, first name, last name, email, role, and password. The password is securely hashed using Argon2 encryption.

Next, **Route** collection (or simply pipeline) is the central part of the system and experiences the highest query frequency. As a result, during the design phase, we took the efficiency of getting the relevant information for a route into consideration when making the class diagram and structuring the database. In the end, the Route collection stores ID, name, description, line_points, and markers. Markers are represented as a list of object IDs referencing corresponding markers. Furthermore, **Marker** collections contain id, name, description, latitude, and longitude, and **Survey** collections contain id, name, depth, notes, created at date, dGPS data, and marker reference (marker's id).

Lastly, **Session** collections are created to back up the session data of the user in case of a power outage and server failure. It contains ID, expiry date, and session object which contains user information, cookie information, and login status. This collection is automatically created and managed by the "connect-mongodb-session" library.

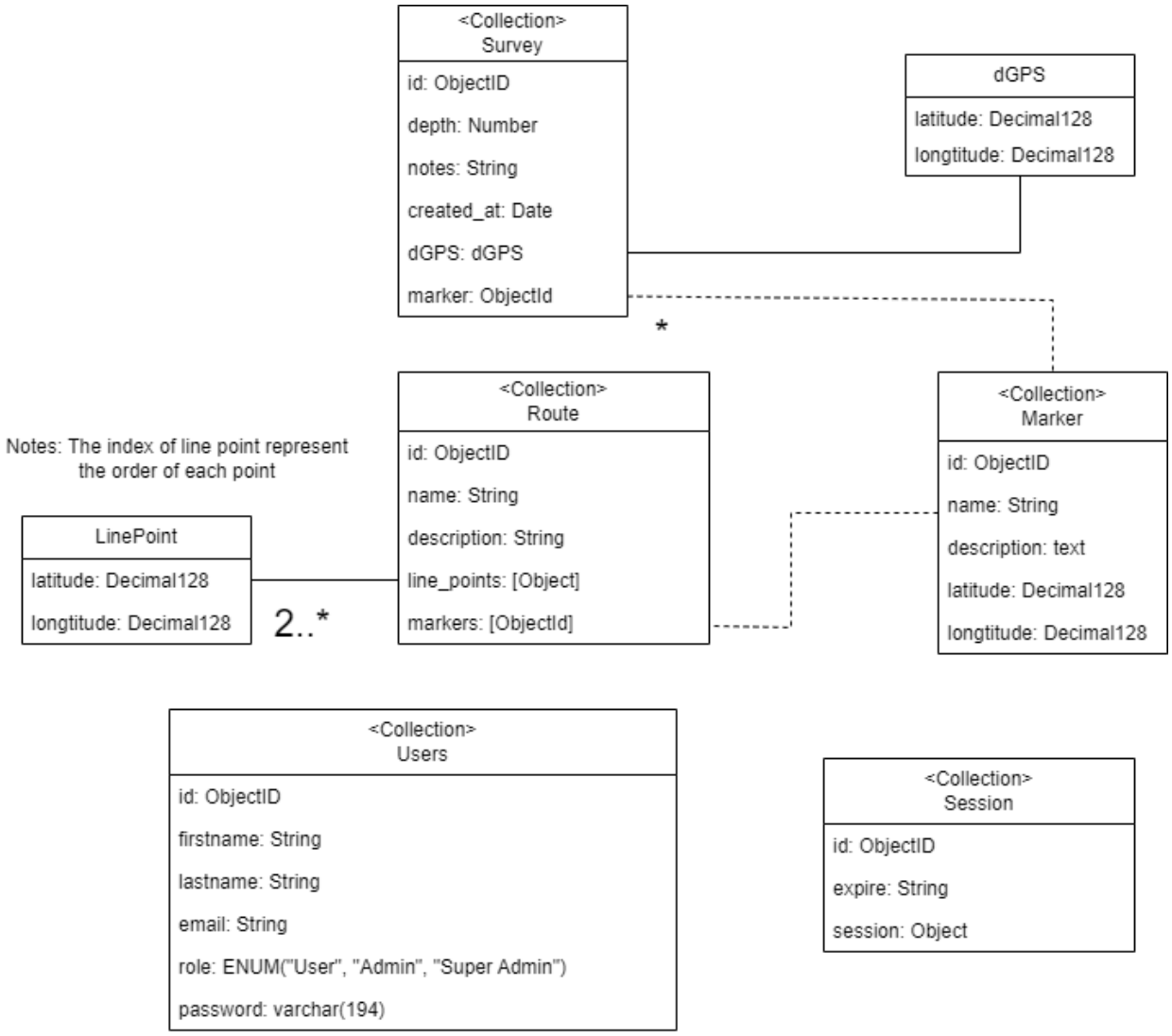


Figure 5.1: Database diagram representing the storage structure in MongoDB

One interesting choice of database structure that we would like to discuss is the decision to separate markers and routes into different collections. There are two main reasons for doing this. The first reason is that a route can contain a large amount of information, which may exceed the limitation of MongoDB. MongoDB has a maximum document size of 16 MB, and separating markers from routes helps prevent this limitation from being exceeded.

Another reason is to eliminate the need for referencing pipelines in the survey collection. This is because when generating an ID in a subdocument (particularly when the marker object is inside a pipeline document), the ID is only unique within the subdocument layer. Therefore, storing survey information alongside pipeline IDs would result in longer URLs to accommodate the pipeline IDs.

The disadvantage of this change is that it takes a longer time to query for marker information. However, that does not post a problem, since we require all the pipeline information before we query the related marker information.

6 Testing The System

To develop a robust application that won't break under real-world conditions we make sure to thoroughly test our system. Tests have been conducted on both the front-end and back-end of the system to ensure proper implementation.

As we used an agile approach in this project, requirements were added and changed iteratively during the development of the system. Unit tests and automated tests were created every time a feature was added to the project, existing tests were also changed if necessary. Before any of these new functionalities were added to the project's code base, all tests had to pass to verify correctness and confirm no previously implemented functionality had broken.

6.1 Test plan

The used testing approach includes a combination of manual testing and automated testing. Automated tests will not achieve 100% coverage but rather focus on covering all high-traffic scenarios. These tests were split into two main categories, the back-end unit testing and the front-end integration tests.

We utilized CI/CD for automated testing by running the full test suite every time a new section of code was committed to the development branch. This ensures that the code written not only works in the local environment but also in a controlled environment.

Parts of the system that were not tested automatically were tested manually. In these manual tests, we navigate through the application mimicking real-world scenarios. While manual testing has limitations such as human error and scalability, it allows us to quickly explore and validate the functionality of the application, aligning the duration and the goal of the project.

6.2 Defined tests

In this section, we will give an overview of the automated tests that have been created to ensure the proper functionality of the application.

6.2.1 Back-end unit tests

The backend testing employed unit tests for every API endpoint. Supertest and Jest were used because they simplify API testing, can do fast and parallel testing, and Jest has a built-in coverage report so that the team can easily analyze the test cases.

There are many benefits to applying unit tests. Because there are multiple API endpoints in the system, ensuring every endpoint is working properly is important. Unit testing can help the team ensure all the endpoints are working correctly. Our unit tests mock API calls to the system with different payloads and verifies that the given response is correct. Furthermore, unit testing ensures the team can modify the APIs and code at any time without breaking existing functionality.

APIs to be tested:

Profile API

1. *GET Profile*
2. *UPDATE Profile*

Users API

1. *CREATE A New User*
2. *GET All Users*
3. *DELETE User*
4. *GET User by ID*
5. *UPDATE User by ID*

Auth API

1. *LOGIN*
2. *LOGOUT*

Pipeline API

1. *CREATE A New Pipeline*
2. *GET All Pipelines*
3. *DELETE Pipeline*
4. *GET Pipeline by ID*
5. *UPDATE Pipeline by ID*

Marker API

1. *CREATE A New Marker*
2. *DELETE Marker*
3. *UPDATE Maker by ID*

On-site Survey API

1. *CREATE A New Survey*
2. *GET All Survey by Marker*
3. *DELETE Survey*
4. *GET Survey by ID*
5. *UPDATE Survey by ID*

The team makes test cases to test every API to ensure all the API functions are covered and pass the test with the correct functionalities.

6.2.2 Front-end integration tests

The front-end testing mainly consists of automated simulations of user input for high-traffic situations. Examples of high-traffic situations are login and logout, managing users, and viewing and editing pipelines or markers.

Specific tests that have been automated are listed below. Any functionality of the system that is not listed here has still been tested manually but has not received an automated test for better validation. Functionality that has not received an automated test has either been deemed non-critical to system functioning, or one of the automated tests already requires this functionality to work, indirectly testing this functionality.

Login and logout test

The login and logout test simulates different user inputs on the login page as well as simulates a click on the logout button. The tests run through the following scenarios and check if the expected response was shown to the user:

1. When the user leaves the email address field empty, an error message is displayed telling the user they are missing an email address.
2. When the user enters an invalidly structured email address, an error message appears telling the user the email address is incorrect.
3. When the user leaves the password field empty, an error message is shown telling the user they are missing a password.
4. When the user enters an incorrect email and password combination, the user should get notified that an account with the given email and password combination does not exist.
5. When the user enters the correct email and password combination, the user should be logged in and redirected to the map page.

Users test

The user test simulates the creation, viewing, editing, and deleting of users in the system. The test will run through the following scenarios and verify if the given response is correct:

1. When a new user is created, the user should be displayed in the list of users.
2. When the details of a user (email, first name, last name, role) are edited, a reload of the page should display the updated data.
3. When a user is deleted, the user should be removed from the user list.

Pipeline test

The pipeline test simulates the creation, viewing, editing, and deleting of pipelines in the system. This test will go through the following scenarios and validate if the response is correct:

1. When a new pipeline is created, the user should be able to select it on the map page resulting in the pipeline being displayed.
2. When the details of a pipeline (name, description) are edited, after a refresh of the page, the pipeline's data should reflect the changes made.
3. When a pipeline is deleted, the user should no longer be able to select the pipeline on the map page.

Marker test

The marker test simulates the creation, viewing, editing, and deleting of markers on a pipeline. This test will run through the following scenarios and verify if the results are as expected:

1. When a new marker is created, the user should be able to click on this marker when the correct pipeline is selected, resulting in the data of the pipeline being displayed.
2. When the details of a marker (name, description) are changed, a reload of the page and re-selection of the marker should display the updated details.
3. When a marker is deleted, the user should no longer be able to see the marker on the map.

6.3 Unit Test Coverage

All APIs have been tested to ensure that the correct information is displayed and that the appropriate HTTP response is returned for Super Administrators, Administrators, and Users.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	84.64	62.5	81.65	87.9	
api	85.41	16.66	33.33	85.41	
app.js	85.41	16.66	33.33	85.41	29,48-49,91-96
api/helpers	72.41	62.5	100	72.41	
db.js	80	100	100	80	14-15
errorHandler.js	64.28	58.33	100	64.28	4,8,12,21,29
model.js	75	50	100	75	5
role.js	100	100	100	100	
api/middleware	80	75	100	80	
session.js	80	75	100	80	7,16,22
api/models	100	100	100	100	
marker.js	100	100	100	100	
pipeline.js	100	100	100	100	
survey.js	100	100	100	100	
user.js	100	100	100	100	
api/routes	82.38	59.67	76.31	83.91	
auth.controllers.js	86.36	66.66	83.33	95	24
index.js	80	100	0	80	6
marker.controllers.js	80.76	50	66.66	80.76	51,62,74,80,91
pipeline.controllers.js	80.48	42.85	81.25	80.48	66,75,86,101,123-126,142
profile.controllers.js	90	75	71.42	89.47	53,63
survey.controllers.js	86.48	83.33	75	86.11	78,86,94,115,128
user.controllers.js	77.96	60.71	80.95	80.76	64,84,93,101,105,111,120,133,152-154
api/services	88	67.85	100	98.18	
marker.services.js	85.29	50	100	100	14-66
pipeline.services.js	92	66.66	100	100	64-82
survey.services.js	89.28	62.5	100	100	40-70
user.services.js	86.84	75	100	94.28	42,65

Figure 6.1: Test coverage result for unit testing

With all unit tests combined, as shown in Figure 6.1, we reach coverage of 87.9% of all lines for the backend part of the system. This is a satisfying result considering that the majority of the uncovered lines and functions are to handle unforeseen or last resource errors. Overall, these test results provide a comprehensive assurance that all the API functions perform as expected in reference to the documentation.

7 Evaluation

This chapter evaluates the development process of the project by the team. First, the planning phase of the project is reviewed. Then, the responsibilities of each member are listed. Furthermore, how the team communicates is discussed. Finally, the result of this evaluation and conclusion is given.

7.1 Planning

At the beginning of the project, the team decided to use the Scrum framework for product development. We set up 5 sprints, with each sprint lasting two weeks, except for the first and last sprints. At the end of each sprint, we would have a sprint review meeting with our client to showcase the developed features and obtain feedback. Every Friday, we would have a meeting with our supervisor to ensure we were on the right path. This predetermined schedule helped us to set milestones for each iteration of the project with a clear deadline, allowing us to keep track of progress and spot any mistakes in our planning early on with plenty of time to adjust.

As discussed in the Project Management section, the user stories that contained the work that was to be done were divided into sprints and put onto a JIRA board. This board gave us a good overview of work that would be done in a specific sprint, the people assigned to specific tasks, and also the implementation progress of these features.

7.2 Responsibilities

To have a clear separation of responsibilities between our team we assigned specific roles to each member. Although we would all be responsible for the final product and thus helped each other where necessary, this role division allowed us to more effectively assign work to specific team members. The roles and responsibilities for those roles were defined as follows.

Tintin Wongthanaporn (s2712911)	Responsible for back-end implementation. This included handling authentication and authorization, implementing, testing, and documenting the API, and implementing security measures. Moreover, he is responsible for setting up and organizing file structure for the back-end for ease of collaboration and readability of the code. In addition to back-end implementation, he also contributes to front-end implementation, particularly survey management and sidebar functionality (with Jason).
Yixuan Zhuang (s2200848)	Responsible for back-end testing and front-end implementation. This included developing the sidebar functionality which shows the information of the selected markers and pipelines. Furthermore, using Jest and Supertest to create test cases for unit testing the back-end API endpoints(with Tintin), ensures all the API endpoints work correctly.
Jump Srinualnad (s2690837)	Responsible for front-end implementation. Initially, I ensured Leaflet integration into our project, setting it up to display the map and interact with user actions. I then focus on pipeline management (with Jason) and marker

	management. Pipeline management includes making sure that the new pipeline is created properly and that all functionality works. Maker management enables users to place markers on the map intuitively. Making sure marker was designed to contain all required information such as location details. Additionally, I implemented CSV import/export functionality to meet the client's expectations, to enable better data transfer to our application.
Jason Hsu (s2708442)	Responsible for front-end implementation. This included implementing the login page, the profile page, and the users page, implementing pipeline management (with Jump), marker management, cookie management, sidebars (with Tintin), and map settings on the home page, ensuring the cleanliness and consistency of the application, testing frontend manually, and demonstrating the application.
Thijs Beumer (s2761831)	Responsible for setting up initial project repo, creating CI/CD pipelines, managing the JIRA board, and the first line of communication with the client. Main responsibility was making sure the team was aware of progress and knew what to do at all times. Also responsible for creating Front-end tests and assisting front-end development where necessary.

7.3 Team Evaluation

7.3.1 Communication

The team maintained good communication throughout the entire development process via a group chat we made on the platform Discord. Using this channel, we could quickly and efficiently reach our team to give other team members the ability to provide timely feedback and assistance. This allowed us to tackle roadblocks quickly and not let them slow down the process of implementing all agreed-upon features.

7.3.2 Meetings

Our team held several offline meetings every week to help each other solve difficulties encountered and to ensure that everyone was on the right track. Most of the meetings are held in the school library. There, we obtained a quiet environment for discussions and made many key design choices. We found that even though our Discord channel was a good medium to discuss smaller roadblocks and help resolve some ambiguities, in case of larger questions there would sometimes be miscommunication over Discord. The in-person meetings allowed us to have a quicker discussion with the entire team, and also visualize any thoughts in our heads using whiteboards. This often gave a better-aligned view across the team members of how a feature should be implemented.

7.4 Future improvements

At the end of this project, we would like to list down all the possible future improvements that could be made to the current final product. While the client and we are satisfied with the final product, the project's limited duration prevented us from implementing additional features that we will discuss below. These

features were either not included in the initial requirements due to the consideration of time constraints or proposed by the client at the end of the project.

7.4.1 Functional Improvement

While the client is satisfied with the final product, there are some improvements that could be made to ease the user experience and allow usage of the application in more extreme conditions.

First of all, optimizing “optimal marker placement” is one of the essential future improvements. In the current final product, we have the optimal marker placement functionality called “Suggest Marker”. However, the feature only considers the distance between each marker. Therefore, the future improvement of this feature will involve taking terrain accessibility and road accessibility into consideration when suggesting optimal marker placement to users.

Moreover, we would like to implement offline mode functionality if the project lasts longer. It is a feature we prioritized as a "won't-have" in the requirement list because it is not essential compared to other requirements and might even take more time to implement, as offline functionality requires the application to store data locally on the device. Nevertheless, we offer the feature to export the entire pipeline, including the markers placed upon it. This could mitigate the disadvantage of not being able to access the application without an internet connection to a certain extent.

In addition, storing images in the survey collection in the database and displaying them in the application is also a future improvement we could make. It is a new could-have feature that the client recommended during the second meeting. After evaluating the potential time investment required for this feature, we concluded that this feature should be left for future improvement.

Furthermore, we could combine the profile page and user page in the future. Currently, these are two separate pages; the super administrator and administrator can access both pages, while the user can only access the profile page. However, based on the client’s feedback during the second meeting, it would be better if these two pages were combined or had a dropdown in the header to choose between them, as users won’t visit them too frequently. We considered this a nice suggestion, but we were busy with the survey management, and other new requirements proposed during the same meeting and had insufficient time to implement the change. Therefore, combining the profile page and the user page is one of the potential improvements we would like to implement in the future.

Another potential improvement for the future is enabling users to select multiple pipelines. Currently, users can only select one pipeline at a time. This aligns with our design goal of keeping the system clean and simple to use by allowing users to focus on one pipeline at a time. However, during the second meeting with our client, they mentioned the idea of a new could-have feature allowing users to select multiple pipelines. While this would enhance the functionality of the application, transitioning from selecting only one pipeline to selecting multiple pipelines would require significant changes to the codebase and the addition of related functionalities. For instance, in the Add Marker mode, users can currently add suggested markers to the selected pipeline. If users were able to select multiple pipelines, they would need an additional popup to choose which pipeline to add the markers to. Therefore, we decided to prioritize other new requirements from the client, such as survey management, which are

deemed more critical than the ability to select multiple pipelines. Nonetheless, enabling users to select multiple pipelines remains a valuable potential improvement for the future.

7.4.2 Scalability Improvement

As the nature of the project is a proof of concept, there are many potential scalability issues to be addressed for further improvement.

One of them is that the Rosen employee would have a hard time selecting the correct pipeline using the dropdown in our current design as the number of pipelines increases in the system. To mitigate this potential issue, there are two main approaches that can be considered for future improvement.

First, implementing a filterable dropdown menu would allow employees to enter the name of the pipeline they are looking for, narrowing down the options displayed. However, this may not solve the problem entirely since it still requires employees to remember the name of the pipeline which may not be feasible as the number of pipelines expands.

Secondly, a more comprehensive solution involving linking and grouping pipelines based on the clients of Rosenxt who hire Rosenxt to inspect their pipeline. With this approach, employees can better navigate and locate pipelines based on the client. The approach aligns with how Rosenxt manages its data and offers a more intuitive solution for pipeline management as the application continues to scale.

7.4.3 Security Improvement

Addressing the security features is not as important as a proof of concept product. However, in the real-world scenario, these issues need to be addressed to ensure that the application meets industry practices.

One of the current standards to meet is the inclusion of Multi-Factor Authentication. Given the project's scope and priorities, implementing this feature was not deemed the most important. Nevertheless, it is definitely a good security feature to prevent someone from accidentally guessing the password correctly and directly logging into the system.

Furthermore, we will greatly include audit logs in the future, which would enable the detection of suspicious activity or unauthorized access attempts. This would be considered a crucial feature to have if the application were to be used in a real-world scenario rather than as a proof of concept.

7.5 Conclusion

We, as a team, are confident that our work aligns with the agreed-upon specifications from our client as a result of our collaborative effort. We had a well-structured plan for the project at the beginning and every team member knew what to do. The communications between team members went well and we were happy to help and to cooperate with each other. During the project, we gained knowledge and practiced skills for creating a web application. Overall we look back on a successful project and are happy with what we have been able to create for our client.

Bibliography

[1] A. Srivastava, S. Bhardwaj and S. Saraswat, "SCRUM model for agile methodology," *2017 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, 2017, pp. 864-869, doi: 10.1109/CCAA.2017.8229928.

[2] Atlassian (2023), *JIRA Software (Version 9.12)* [Computer software], Retrieved March 15, 2023, from https://www.atlassian.com/software/jira

[3] Lile, E. A. (1993). Client/Server architecture: A brief overview. Journal of Systems Management, 44(12), 26. https://www.proquest.com/openview/6f660e1578ee4f9741d31bf0bb2aef3f/1?pq-origsite=gscholar&cbl=40682

Appendix A - User Manual

Set up

Installation

1. Use Node Version Manager (nvm) to install and use node 20.11.0. Install it with "nvm install 20.11.0" and then switch to it using "nvm use 20.11.0".
2. Install all dependencies using “npm i”.
3. Follow the database instructions to install a local database.
4. Set up configuration files according to the configuration file instructions.
5. Run the database seeder to add initial data (the super administrator account) to your database with “npm run seeder”.
6. You are ready to go!

Database

The database used in this project is MongoDB.
Click here to download and follow the instructions in this link.
Use these settings:

- Setup type: complete
- Run the service as a Network Service user and press “**next**”.
- Select “**install MongoDB compass**" and hit “**next**” again.
- Then press “**install**” and wait for the setup to finish.

After this process, you should have both MongoDB (the database) and MongoDB Compass (a GUI for the database) installed on your device.

Configuration file

- Use “cp config.example.json config.json” to copy the example config file.
- Use “npm run server:keygen” to generate server secret
- Use “cp client/src/config.example.json client/src/config.json” to copy the example config file for the client side
- You may change the predefined **connection string** if necessary. You could get the connection string from the previously installed MongoDB Compass application.
    - Connect to your database in MongoDB Compass.
    - Click on the 3 dots next to the name/URL of the database.
    - Then click on "Copy connection string".
    - Finally, paste the copied string into the config.json file.

For further information, please refer to the **README** file in the repository.

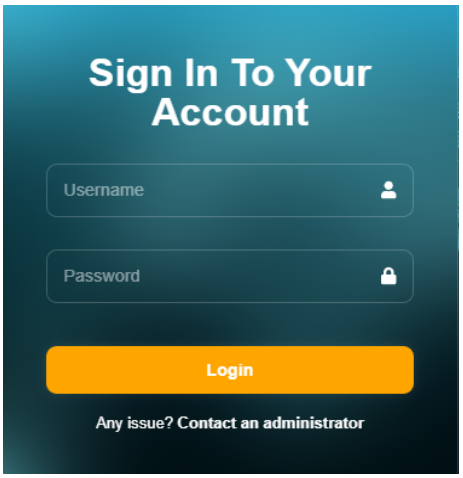
Authorization & Authentication

Roles

The application consists of three roles: **super administrator**, **administrator**, and **user**. Each role has different levels of access to the application's features. Both the super administrator and administrator have access to all functionalities. The only difference between them is that the super administrator can create administrators. On the other hand, the user will have limited access to the application's functionalities.

Throughout this user manual, sections that do not specify an allowed role indicate that the feature is accessible to all roles, whereas those specifying a particular role are limited to that role alone.

Login



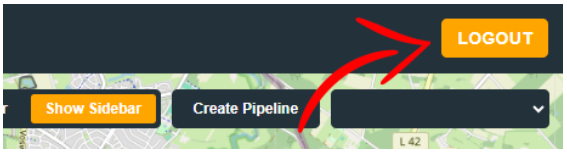
To log in to the application, please enter your username, which should be an email address and the password.

Important notes:

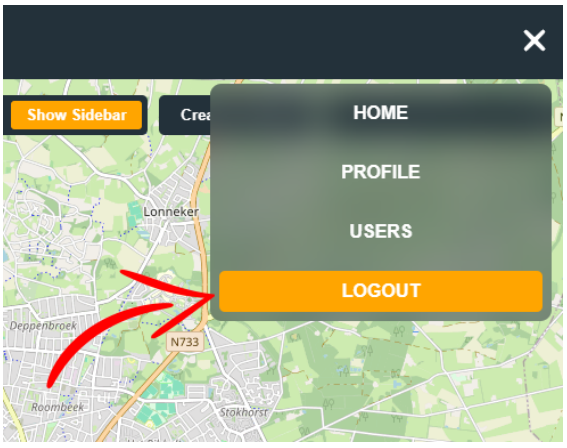
- The password must be between 8 and 30 characters long.
- There is no 'forgot password' functionality. If you encounter any issues with your account, please contact a (super) administrator directly if you have a user account, or a super administrator if you have an administrator account.

Logout

The **Logout** button is on the header of the application.

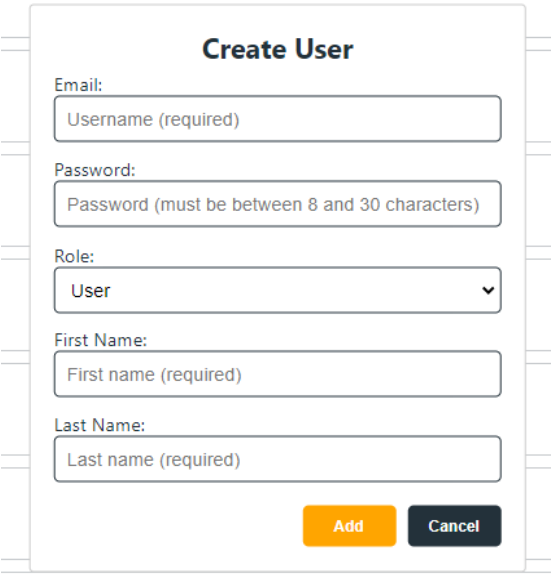


If the screen size is smaller, the **Logout** button will be located at the bottom of the **dropdown menu**, as illustrated in the right image.



Account management

Create an account



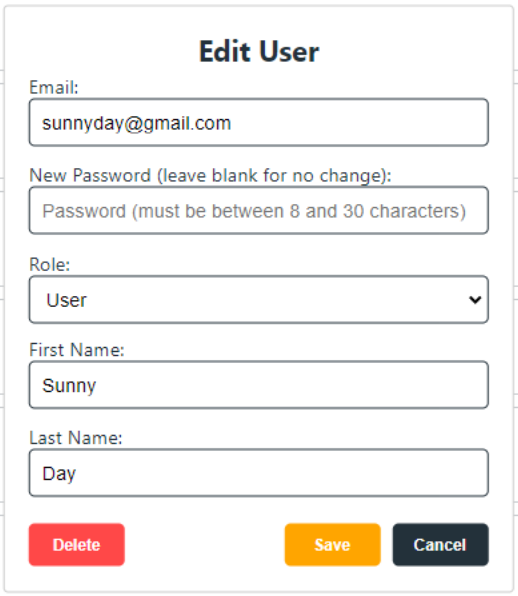
Role: super administrator/administrator
Steps:

1. Go to the **User** page.
2. Click the **Add** button on the top right corner.
3. Make sure every field in the **Create User** modal is filled out (all fields are required).
4. Create the user by clicking the **Add** button at the bottom of the modal.

Important notes:

- The super administrator can create administrator accounts or user accounts while the administrator can only create user accounts.
- The email (username) should be unique.
- Passwords must be between 8 and 30 characters.

Edit an account



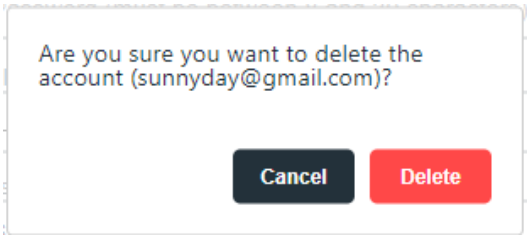
Role: super administrator/administrator
Steps:

1. Go to the **User** page.
2. Click the **Edit** button on the right side of the account row.
3. Make sure all fields except the **New Password** in the **Edit User** modal are filled out.
4. Click the **Save** button to update the account.

Important notes:

- The super administrator can edit administrator accounts or user accounts while the administrator can edit user accounts.
- If you don't want to change the password, leave the **New Password** field blank.

Delete an account

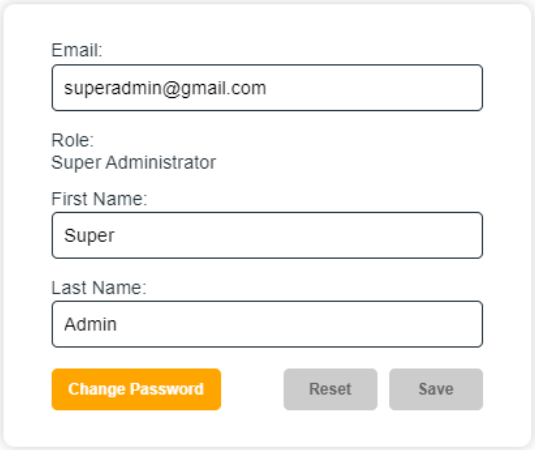


Role: super administrator/administrator
Steps:

1. Go to the **User** page.
2. Click the **Edit** button on the right side of the account row.
3. Click the **Delete** button in the bottom left corner of the **Edit User** modal.

4. Click the **Delete** button again in the **Confirmation** popup.

Update own profile



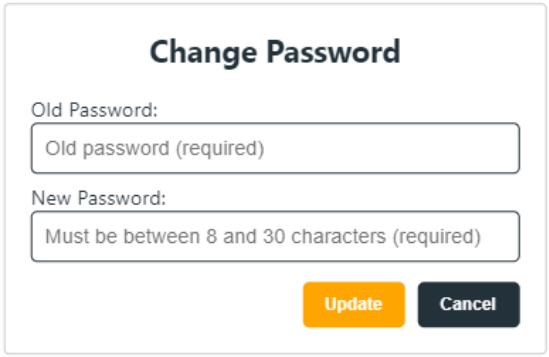
Steps:

1. Go to the **Profile** page.
2. Update the fields to the new value.
3. Click the **Save** button to update the profile.
4. Click the **Reset** button if you want to discard the current change.

Important notes:

- The account's role cannot be changed by its owner.
- The email (username) should be unique.

Change password



Steps:

1. Go to the **Profile** page.
2. Click the **Change Password** button.
3. Enter the **Old Password** and the **New Password**.
4. Click the **Update** button.

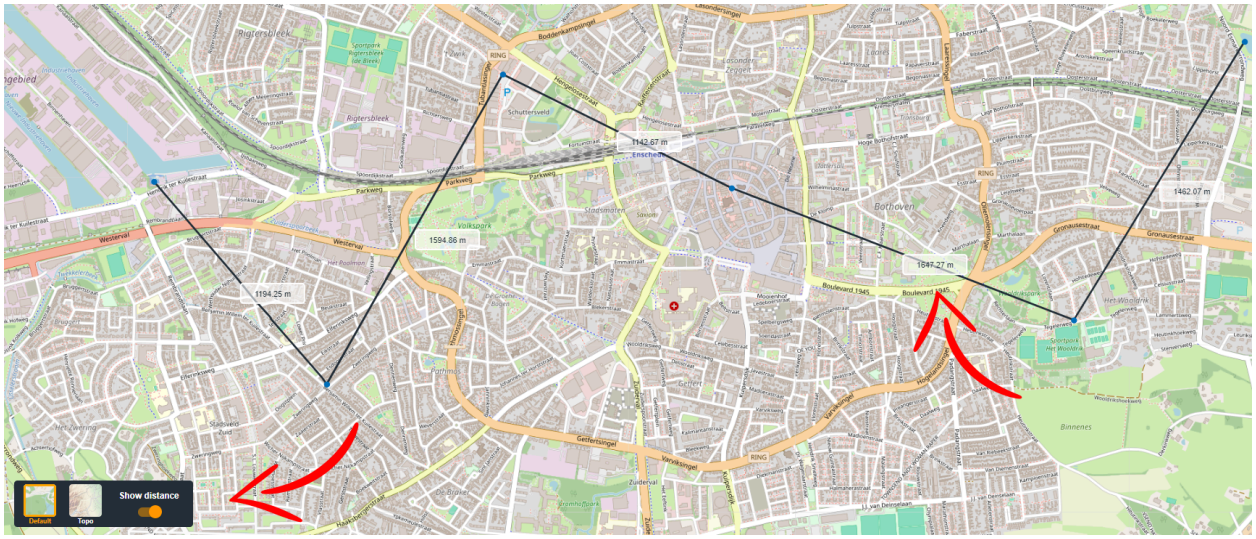
Important notes:

- Both the old password and the new password must be between 8 and 30 characters in length.

Map setting

Open distance label

By default, the distances between markers and pipeline points aren’t displayed on the map. To enable this feature, check the 'Show distance' switch at the bottom left of the **Home** page.



Switch map style

By default, the map style is the **Default** map style, as shown above. To switch to the topographic map, select the **Topo** map style at the bottom left of the Home page.

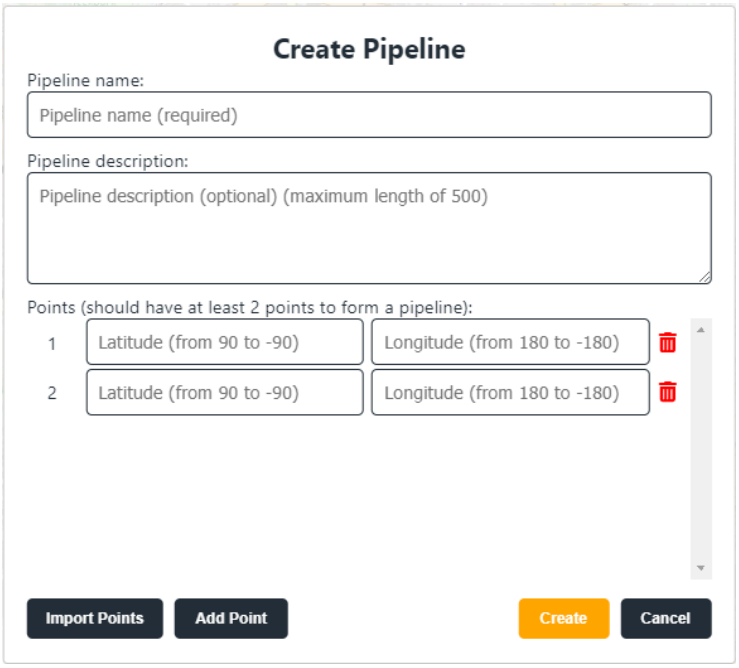
Important note: The topographic map takes a longer time to load.

Pipeline management

Create pipelines

Role: super administrator/administrator
Steps:

1. Go to the **Home** page.
2. Click the **Create Pipeline** button on the top right corner.
3. Make sure all the fields except the **Pipeline description** (optional) in the **Create Pipeline** modal are filled out.
4. Create the pipeline by clicking the **Create** button at the bottom of the modal.



Important notes:

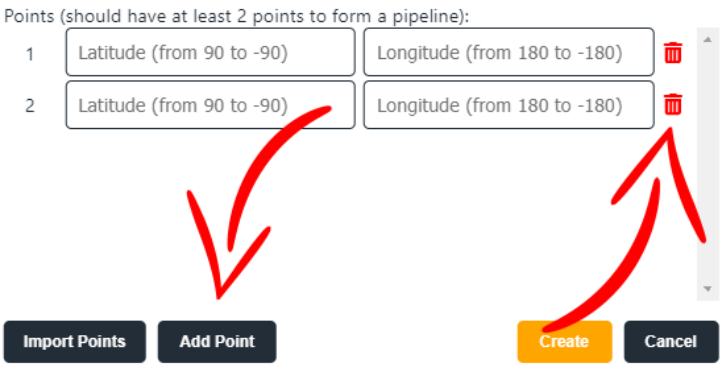
- The **Pipeline name** should be unique.
- The maximum length of the **Pipeline name** is 100 characters.
- The maximum length of the **Pipeline description** is 500 characters.
- The range of the latitude is from 90 to -90.
- The range of the longitude is from 180 to -180.
- To create a pipeline, you need at least two points.

When creating a pipeline, the application offers two methods for adding points (coordinates) to it, as demonstrated below.

Add/Delete point(s) manually

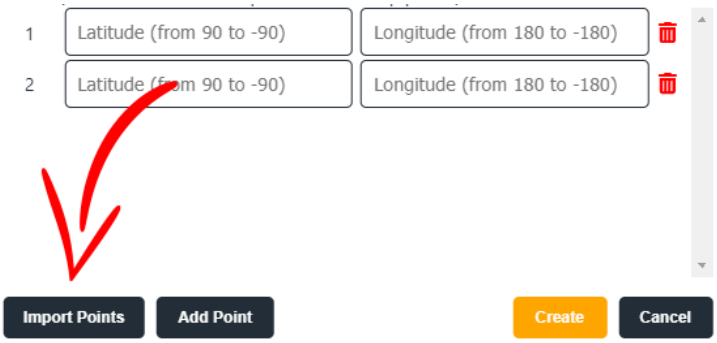
To manually add a point, click the **Add Point** button to insert an empty row into the Points table. Then, you can enter the coordinates.

To delete a point, click the **Trash Bin icon** on the right side of the row.



Import points from the CSV file

To import points from a CSV file, click the **Import Points** button and select the CSV file you wish to import in the file explorer. After importing the CSV file, you still need to provide a name (required) and an optional description for the pipeline to be successfully created.



Important notes:
The CSV file must adhere to a specific format. As depicted in the image on the right, the first row should contain headers, and all subsequent rows should consist of coordinates. The first column should always represent latitude, and the second column should always represent longitude.

	A	B
1	Latitude	Longitude
2	52.22092	6.857164
3	52.21277	6.868579
4	52.22523	6.880166
5	52.22066	6.895187
6	52.21535	6.91776
7	52.22655	6.929004

Switch pipeline

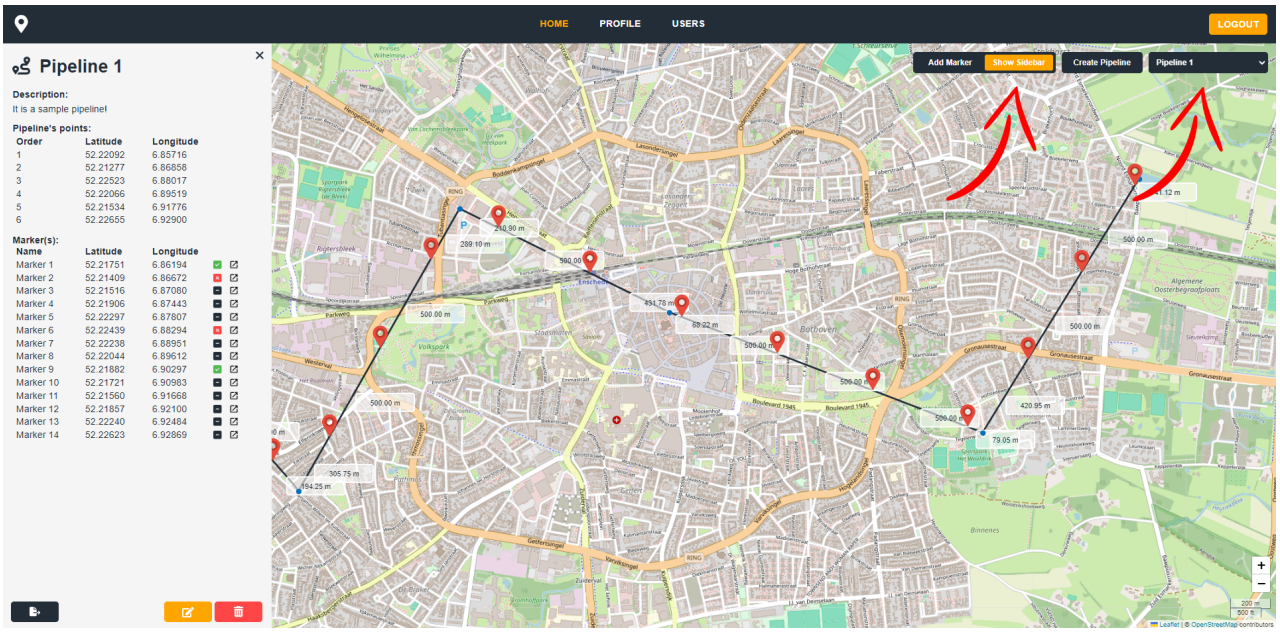
Steps:

1. Go to the **Home** page
2. Select the pipeline from the **dropdown menu** in the top right corner of the page.

Pipeline's Sidebar

Steps:

1. Go to the **Home** page
2. Select the pipeline from the **dropdown menu**.
3. Choose the **Show Sidebar** mode in the top right corner of the page.
4. Click on the pipeline.



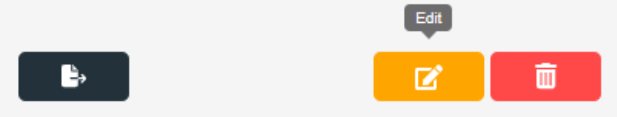
Edit a pipeline

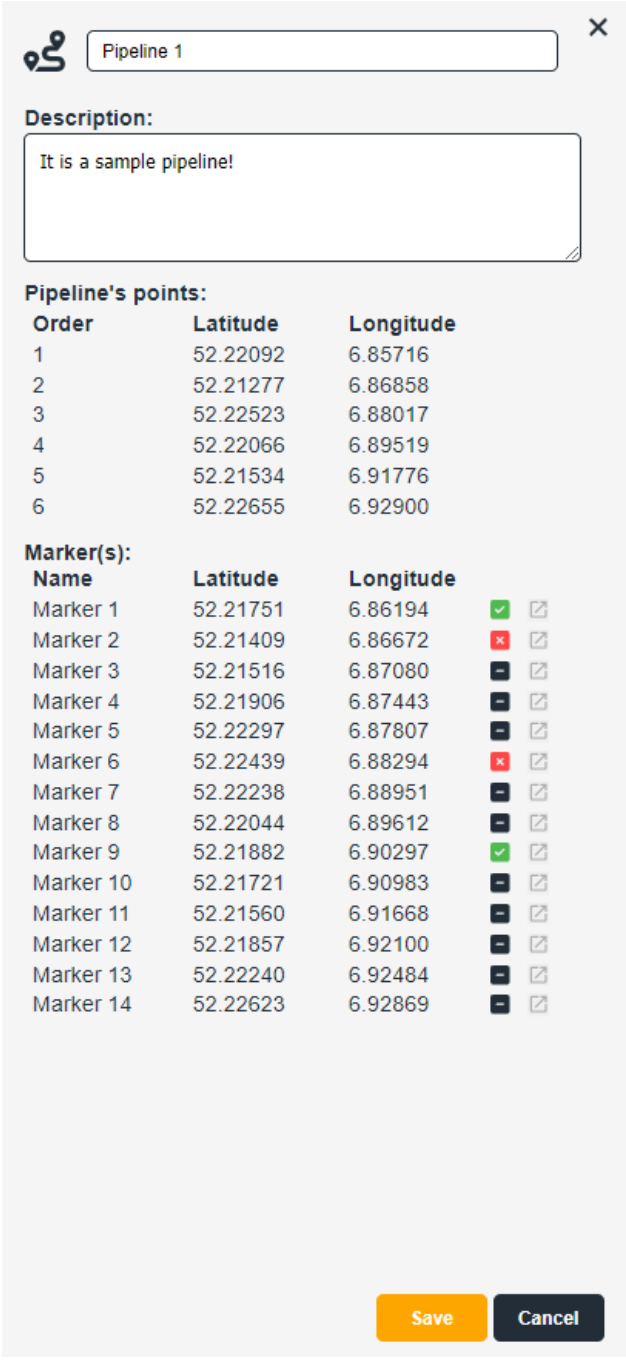
Role: super administrator/administrator
Steps:

1. Go to the **Home** page.
2. Open the sidebar of the pipeline you wish to edit.
3. Select the orange **Edit** icon button at the bottom of the sidebar.
4. Update the fields with the new values.
5. Click the **Save** button to apply the changes.

Important notes:

- The **Pipeline name** should be unique.
- The maximum length of the **Pipeline name** is 100 characters.
- The maximum length of the **Pipeline description** is 500 characters.
- It is not allowed to edit the pipeline's points.

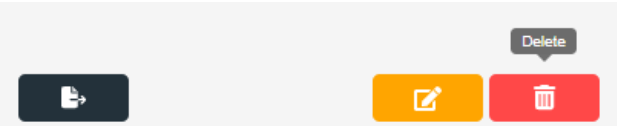




Delete a pipeline

Role: super administrator/administrator
Steps:

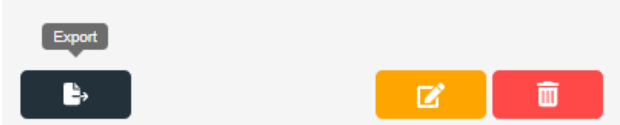
1. Go to the **Home** page.
2. Open the sidebar of the pipeline you wish to delete.
3. Select the red **Trash Bin icon** button at the bottom of the sidebar.



Export a pipeline (including markers on the pipeline)

Steps:

1. Go to the **Home** page.
2. Open the sidebar of the pipeline you wish to export.
3. Click the **Export** button. The exported file will look like the right image.



	A	B
1	Line Points	
2	Latitude	Longitude
3	52.22092	6.857164
4	52.21277	6.868579
9	Markers	
10	Latitude	Longitude
11	52.21751	6.861943
12	52.21409	6.866722

Marker management

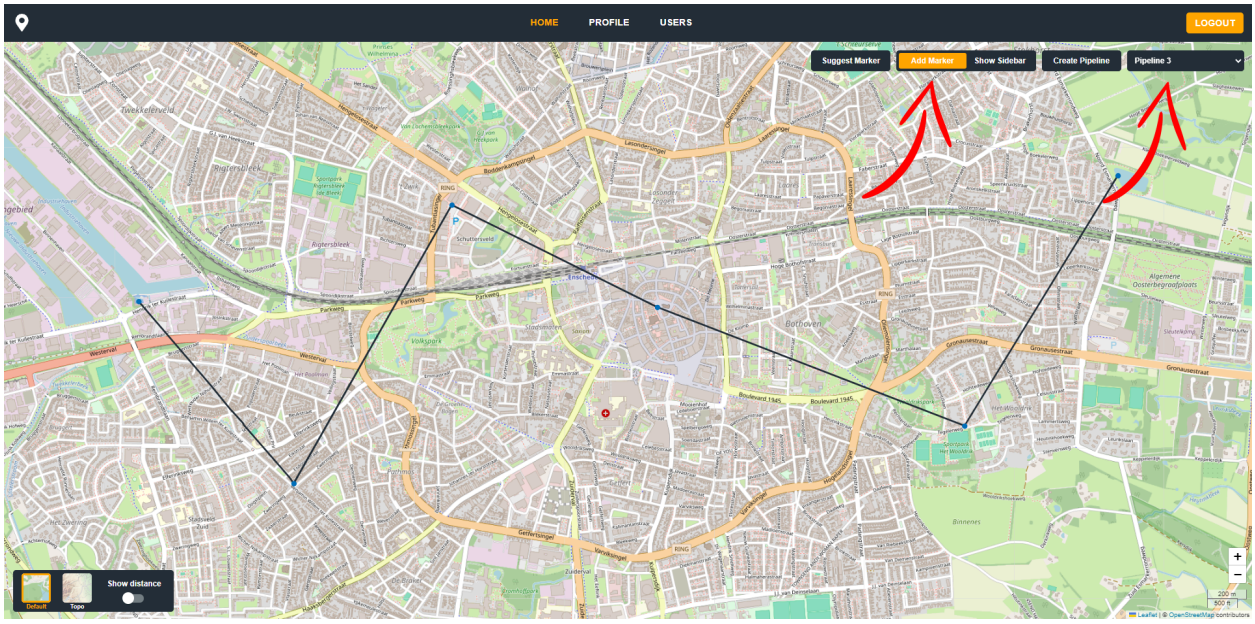
Create markers

Role: super administrator/administrator
Steps:

1. Go to the **Home** page
2. Select the pipeline from the **dropdown menu**.
3. Choose the **Add Marker** mode in the top right corner of the page.

Important notes:

- The application only allows adding markers when a pipeline is selected.
- The maximum number of markers on a pipeline is 500.
- Within a pipeline, there cannot be two different markers with the same coordinates.

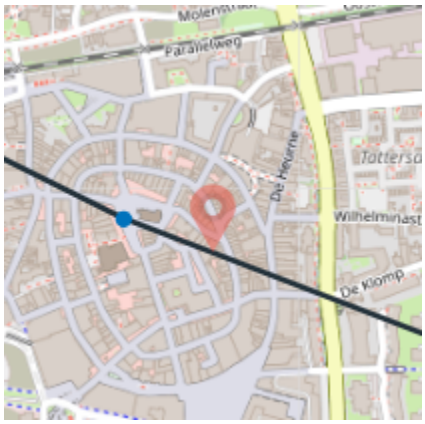


When you are in the **Add Marker** mode, the application offers two methods to add marker(s), as explained below.

Click on the map

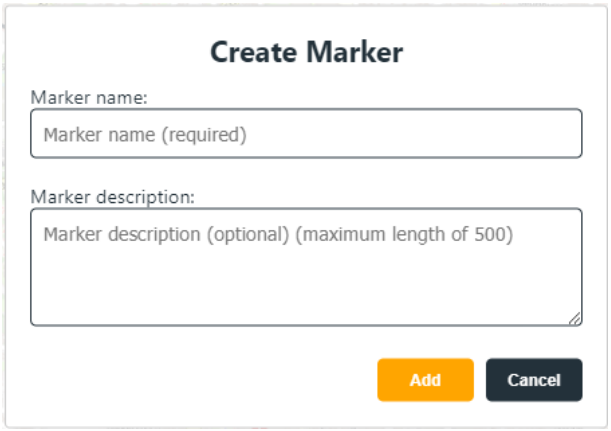
Steps:

1. In the **Add Marker** mode, a predicted marker placement will always be displayed on the pipeline, as shown in the right image. To add a marker, simply click on the map.
2. Enter a marker's name (required) and a description (optional) in the **Create Marker** modal.
3. Click the **Add** button to add the marker to the selected pipeline.



Important notes:

- The maximum length of the **Marker name** is 100 characters.
- The maximum length of the **Marker description** is 500 characters.



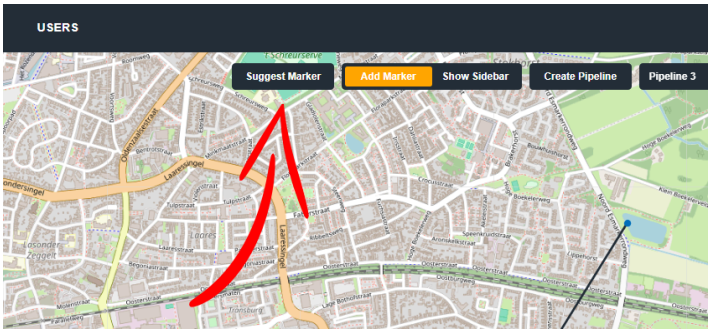
Suggest marker

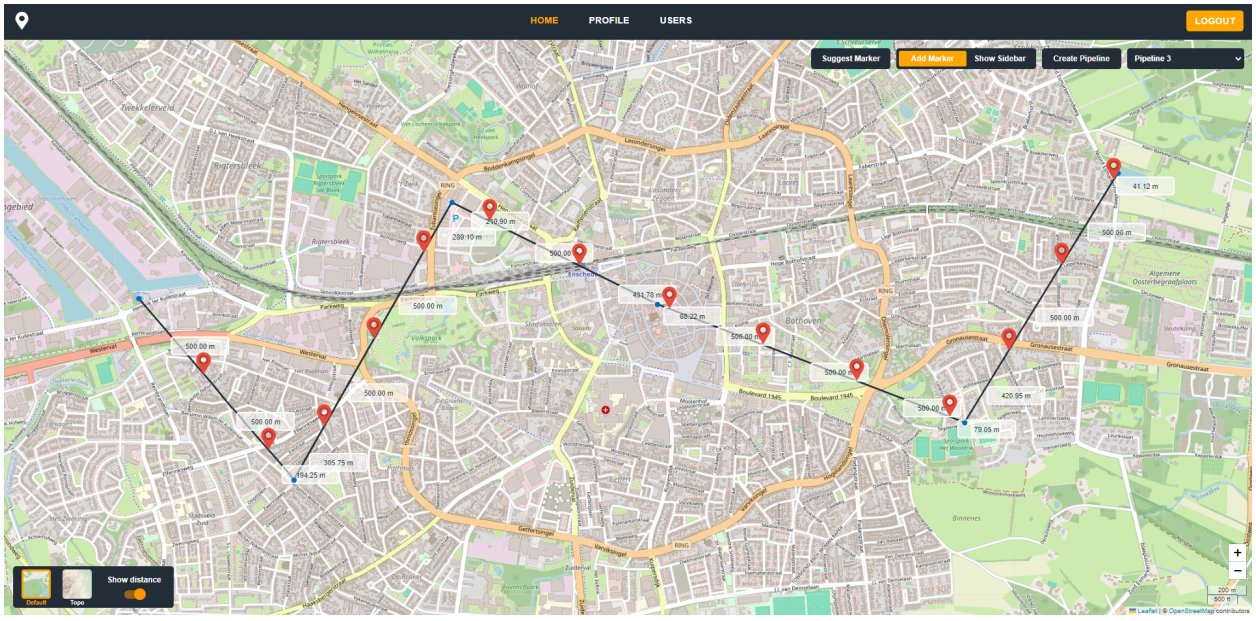
Steps:

1. In the **Add Marker** mode, click on the **Suggest Marker** button, as shown in the image below.
2. Click the **Add** button in the **Confirmation** popup.
3. Markers will be added, each separated by a distance of 500 meters, as shown in the image at the bottom of this page.

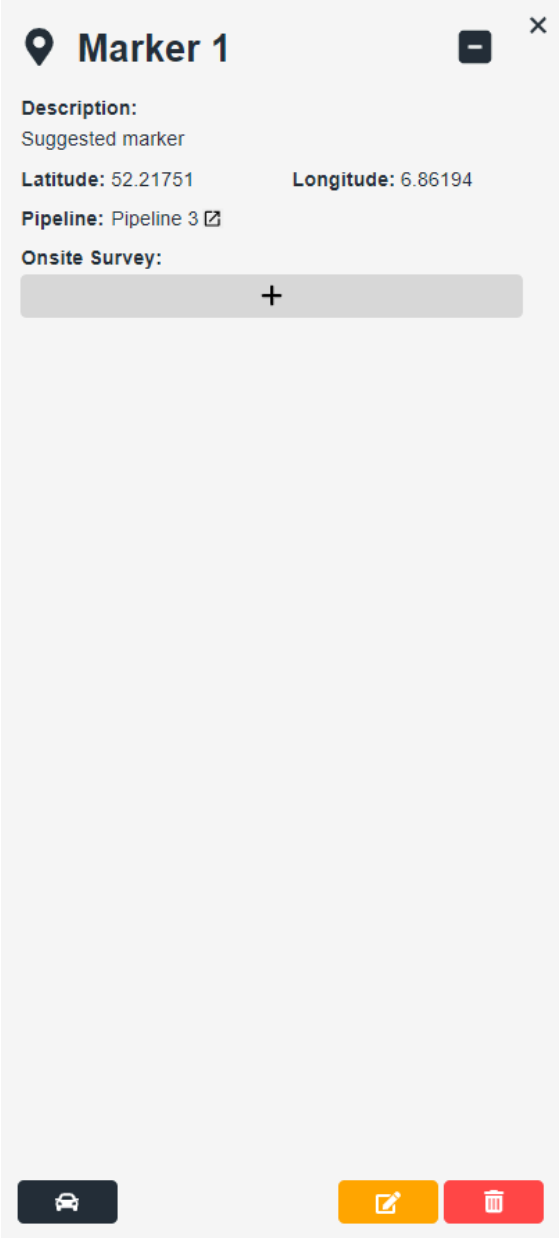
Important notes:

- Each marker will have a default name and description.
- The Suggest Marker function will fail if it results in more than 500 markers after adding.
- The Suggest Marker function will fail if any existing marker has the same coordinates as any markers that will be added.





Marker's sidebar



Steps:

1. Go to the **Home** page
2. Select a pipeline from the **dropdown menu**.
3. Choose the **Show Sidebar** mode in the top right corner of the page.
4. Click on the marker.

Edit a marker

Role: super administrator/administrator

Steps:

1. Go to the **Home** page.
2. Open the sidebar of the marker you wish to edit.
3. Select the orange **Edit** icon button at the bottom of the sidebar.
4. Update the fields with the new values.
5. Click the **Save** button to apply the changes.

Important notes:

- The maximum length of the **Marker name** is 100 characters.
- The maximum length of the **Marker description** is 500 characters.
- It's not allowed to edit the marker's coordinates.

Delete a marker

Role: super administrator/administrator

Steps:

1. Go to the **Home** page.
2. Open the sidebar of the marker you wish to delete.
3. Select the red **Trash Bin icon** button at the bottom of the sidebar.

Navigate to the marker

Steps:

1. Go to the **Home** page.
2. Open the sidebar of the marker you wish to navigate to.
3. Select the dark blue **Car icon** button at the bottom of the sidebar. You will be directed to the **Google Maps interface** with the marker's coordinates, where you can utilize the **Directions** function.

Survey management

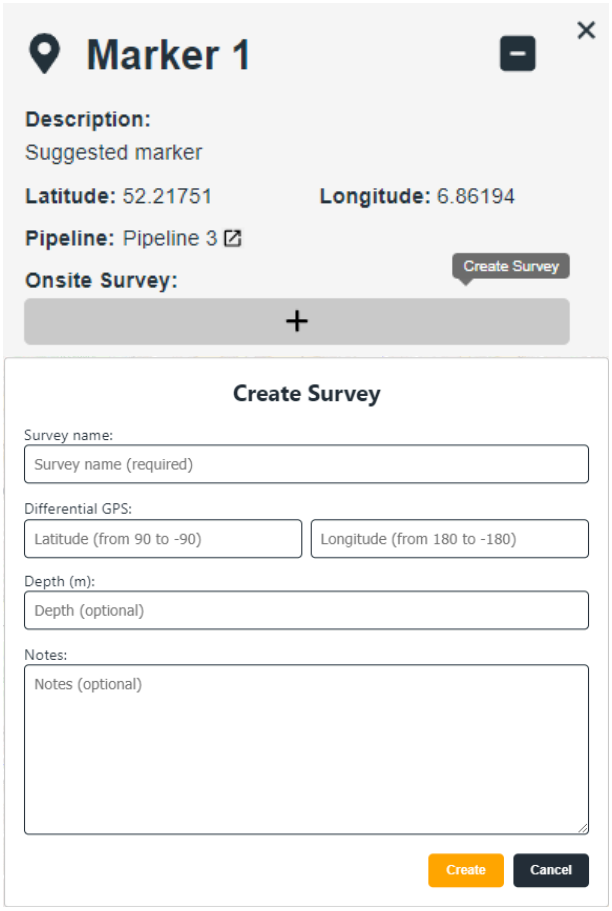
Create a survey

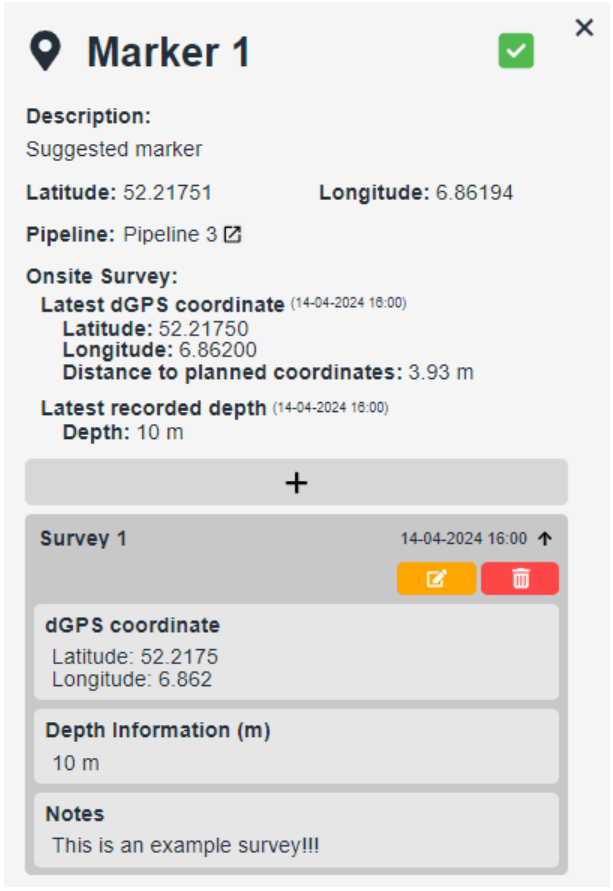
Steps:

1. Go to the **Home** page.
2. Open the sidebar of the marker to which you wish to add surveys.
3. Click the **Plus** icon button under **Onsite Survey**.
4. Enter all required fields, including **Survey name** and **Differential GPS**, and optional fields, if needed, in the **Create Survey** modal.
5. Click the **Create** button.

Important notes:

- The maximum length of the **Survey name** is 100 characters.
- The range of the latitude is from 90 to -90.
- The range of the longitude is from 180 to -180.
- The **Depth** cannot be negative.





Edit a survey

Role: super administrator/administrator

Steps:

1. Go to the **Home** page.
2. Open the sidebar of the marker to which you added the survey.
3. Find the survey under **Onsite Survey** and expand it, as shown in the left image.
4. Click the orange **Edit icon** button.
5. Update the fields with the new values.
6. Click the **Save** button.

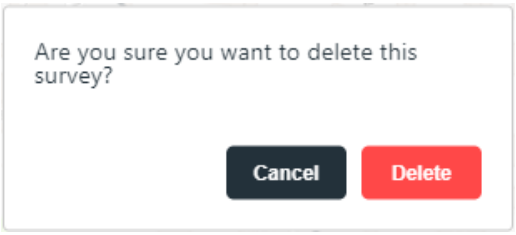
Important notes:

- The same restrictions apply as in the Create a survey.
- The created time of the survey will not change after the survey is edited.

Delete a survey

Role: super administrator/administrator
Steps:

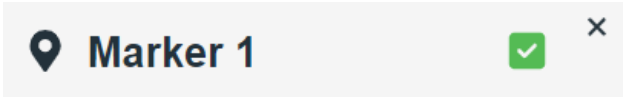
1. Go to the **Home** page.
2. Open the sidebar of the marker to which you added the survey.
3. Find the survey under **Onsite Survey** and expand it.
4. Click the red **Trash Bin icon** button.
5. Click the **Delete** button in the **Confirmation** popup.



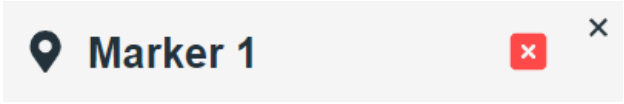
Explanation of the precision icon

For each marker, there will be an icon beside the marker's name in the marker's sidebar.
Three kinds of icons are offered:

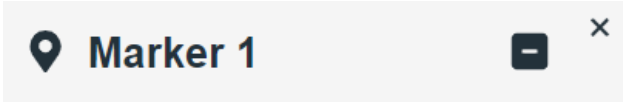
- The **Checked** icon
  (The Checked icon is displayed when the latest dGPS coordinates are within 200 meters of the planned marker's coordinates in the application.)



- The **X-mark** icon
  (The X-mark icon is displayed when the latest dGPS coordinates are outside 200 meters of the planned marker's coordinates in the application.)



- The **Minus** icon
  (The X-mark icon is displayed when no onsite survey has been added.)



Unimportant tips:

- Hovering over the precision icon will show the more accurate distance.
- In the pipeline's sidebar, you can see the precision icon of every marker on the pipeline, as shown in the right image.
- Don't become overly fixated on accuracy because there will always be some degree of error compared to reality due to the curvature of the Earth.

Appendix B - API Specification

Profile API

HTTP method	URL	Description
GET	/api/profile	Get user profile details
PUT	/api/profile	Update a user profile

User API

HTTP method	URL	Description
GET	/api/users	Get details of all users
POST	/api/users	Create a new user
PUT	/api/users/:id	Update an existing user
DELETE	/api/users/:id	Delete an existing user
GET	/api/users/:id	Get details of a specific user

Authentication API

HTTP method	URL	Description
POST	/api/auth/login	Login to the system
POST	/api/auth/logout	Logout of the system

Pipeline API

HTTP method	URL	Description
GET	/api/pipeline	Get details of all pipelines
GET	/api/pipeline/:id	Get a specific pipeline’s details
POST	/api/pipeline	Create a new pipeline
PUT	/api/pipeline/:id	Update an existing pipeline

DELETE	/api/pipelines/:id	Delete an existing pipeline

Marker API

HTTP method	URL	Description
POST	/api/pipelines/:pipelineId/markers	Add a marker to the pipeline
PUT	/api/pipelines/:pipelineId/markers/:id	Update a marker on the pipeline
DELETE	/api/pipelines/:pipelineId/markers/:id	Delete a marker from the pipeline

Onsite Survey API

HTTP method	URL	Description
POST	/api/markers/:markerId/surveys	Create a survey entry for the marker
GET	/api/markers/:markerId/surveys	Get all surveys for a marker
GET	/api/surveys/:surveyId/	Get details of a specific survey
PUT	/api/surveys/:id	Edit an existing survey
DELETE	/api/surveys/:id	Delete an existing survey

Appendix C - Meetings with Rosen

C.1 Meeting 1 - Initial meeting

Feb 29, 2024
The first meeting with the client representative was scheduled for February 29, 2024. Before this meeting, the team had prepared some questions to clarify the requirements and had already emailed the client with some of them.

The meeting took place in Rosen's office in Enschede. The contact person from the client was sick, so he wasn't entirely certain about the project details. We went through the list of questions we had prepared and presented our current requirement list. The representative answered some of the questions, but he needs to confer with his colleagues for answers to the remaining ones. He asked us to email him all unanswered questions and the requirements list and promised to reply to us as early as possible so we could continue with our design and implementation.

Lastly, we reached an agreement to have a sprint review meeting every two weeks at the end of each sprint.

C.2 Meeting 2 (Canceled)

March 14, 2024
Unfortunately due to our client not being able to attend at the time slot we agreed on this week, the meeting was canceled.

C.3 Meeting 3 - Sprint review

March 28, 2024
Due to sickness, our original contact person at the company wasn't able to attend this meeting, luckily a colleague of his was willing to take over. During the meeting, we showed our current progress, which at the time was the full user management system in addition to being able to add and view pipelines and markers. The client assured us that the current product was already in a Minimum Viable Product (MVP) stage and we discussed possible further optimizations and functionalities that could be added. The main points taken from this meeting were that the client would like to have the ability to add on-site survey information such as differential GPS coordinates and depth, for better and more accurate location of markers. Some other minor points such as being able to view multiple pipelines at the same time and being able to suggest optimal marker placements were discussed. In the end, we decided on implementing the on-site survey information as well as making an initial version of the optimal marker placement where "optimal" meant that the markers would be spaced out by 500m.

Overall this meeting was very informative and gave us some good points to further improve our product on.

C.3 Meeting 4 - Final meeting

April 11, 2024
In the final meeting with our client, we gave a demo of the full system, going through all of the functionalities we had added throughout the project. After the demo along with some further clarification for questions the client had after the demo, the client stated to be very happy with the final version of the product. The second discussion point of this meeting was how the handover of the product would look. We decided on handing over the full codebase as well as the design report, which would include a user manual as an appendix. This handover would be done on April 19th, the same time as we would hand in the project to our supervisor.

This concluded the meetings with our client, we thanked the client for allowing us to do this project and agreed to be in touch via email if any further action from the client was needed for our grading process.

Appendix D - Meetings with the Supervisor

This appendix contains a small summary of the meetings between our project team and the supervisor.

D.1 Meeting 1 - Initial meeting

February 14, 2024
The main topic for the first meeting was to introduce ourselves and describe the project to our supervisor. We discussed mainly the contents of the project, how we could best approach the start of our project, and requirement specifications with our client, and finally scheduled weekly meetings until the end of the project.

D.2 Meeting 2

March 1, 2024
This second meeting was right after the first (in-person) meeting with our client. During the meeting, we looked into the way we were applying the scrum methodology alongside the JIRA tool we used for keeping track of our progress and planning future work. The main takeaway from this meeting was that we should make sure the user stories we define would be incremental tangible parts of the project. Using this we could make sure that for the bi-weekly meetings with our client, we would be able to show a demo of the current phase of the project.

D.3 Meeting 3

March 11, 2024
In the third meeting, we showed the proposed structure of our Design Report (this report). Our supervisor suggested some small structural changes which we applied. We also briefly discussed our current progress and views on how communication was between us as a team. One takeaway from this discussion was that we structured our JIRA board in an inefficient way. We divided all the work into back-end and front-end user stories and split those up over the sprints. However, our supervisor commented that it would be better to have actual tangible user stories on our JIRA board and describe all subtasks such as back-end and front-end work within that user story. We chose to change to this approach as it would better show us the progress from the view of what our client could perceive.

D.4 Meeting 4

March 15, 2024
Just before the fourth meeting, we should have had a meeting with our client but unfortunately, this was canceled. As we also had to wait upwards of one and a half weeks for a reply when we mailed our client, we asked our supervisor how to continue with our project. Our supervisor had told us that if we weren’t able to get a reply from the client, we should try and answer our questions ourselves. Although it might be hard to answer some of the questions and our answers are likely to not align with the opinions of the client, we have a deadline to adhere to and thus have to continue working. Apart from discussing the

communication with our client we also demonstrated our current designs in Figma (an online tool for creating user interface designs) alongside our restructured JIRA board, as discussed in the last meeting.

D.5 Meeting 5

March 22, 2024
The fifth meeting was mostly centered around scheduling our final presentation. As of that meeting, our final presentation was scheduled for April 17th during lunch break. During this time slot, there would also be different project groups from our module presenting for the same board. Just before this meeting, we had sent our last revision of the project proposal, after looking into this we concluded that there should be two minor changes. The first was to include stakeholders in the proposal and the second was to change the requirements we added from system requirements to user requirements, to more directly translate from the specific wishes of the client.

D.6 Meeting 6 (Canceled)

March 29, 2024
March 29th unfortunately fell on Good Friday, which is a national holiday in the Netherlands on which the University is closed. Due to this, the meeting we had scheduled was canceled.

D.7 Meeting 7

April 5, 2024
The must-have and should-have requirements that we had implemented were demonstrated to the supervisor. The supervisor provided some useful feedback for the application. Specifically, for the suggested marker placement function, the supervisor suggested that we could have a configuration file to let the users decide what distance between the markers they want instead of a fixed 500 meters. Overall, the supervisor was happy with the progress we had made. We also asked for feedback on the design report and we were suggested that we should have a reflection section in the report to show what we have learned during the project development.

D.8 Meeting 8

April 12, 2024
In this final meeting with our supervisor, we showed the final version of the application and went over our design report to get some initial feedback. Our supervisor suggested some structural changes for the report that we could work on, for example, some of our initial sections were not balanced well or had too deep of a structure making it difficult to understand. Some other minor points of improvement were also discussed such as not naming our sections as “Chapters”. Overall a very informative meeting that we could use to further improve our report.

Appendix E - Sprint Reports

E.1 Sprint 0 | Start-up

Feb 5 - Feb 14
Based on the project description provided by the client, we outlined a set of user stories and requirements and selected the technology stack for building the web application. We contact the client via email and find a supervisor based on both the project's domain and recommendations from the module coordinator. During this sprint, we also created a project proposal, covering planning, risk analysis, and responsibility, although it is yet to be finalized.

In terms of technical infrastructure, we initialized a GitLab code repository and established a set of guidelines for branch management, including creating and merging branches. Lastly, we created a Jira board to organize and prioritize requirements and tasks, which were allocated to each sprint based on their priority.

E.2 Sprint 1 | Design

Feb 15 - Feb 28
The team held a meeting to design the basic layout and functions of the system. Based on the user stories and requirements created last sprint, the team used Figma to visualize the interface of the application. Then, based on the design of functions, the team created tasks on the Jira board. Each of us chose tasks specified under Sprint 1 on the Jira board. Due to some questions about the client's expectations and the ongoing wait for the client to provide clarification, we primarily focused on the absolutely necessary requirements in this sprint.

Frontend The Login page and the Main page have been developed. The login functionality on the Login page integrates with the backend. On the Main page, the layout has been defined. The header with a responsive menu was made, the map was displayed, and basic functionalities to the map, including adding/deleting markers, a sidebar with information about each marker, swapping terrain, etc, were implemented.

Backend The CRUD APIs for "user" and "log in/out" have been created, encompassing the implementation of role management, password encryption, and session management.

E.3 Sprint 2 | Development

Feb 29 - Mar 13
After had the answers to the questions we asked in the last sprint from the client, we started focusing on developing the required must-have functionalities. In this sprint, we mainly focus on the authentication and user management system.

Frontend We finalized the design of the login page. The Users page, where users can see all the users' accounts and their details, was developed. The layout was determined so that the logged-in admin users could easily manage all the accounts, such as adding new users, editing existing users, and deleting users.
Backend API endpoints for user management were created and tested. This includes creating new users, editing existing users, requesting information about users, and deleting users. Also finalized CRUD API for Authentication, Pipeline, and Marker.

E.4 Sprint 3 | Development

Mar 14 - Mar 27
In this sprint, we continued on the rest of the must-priority requirements and started with should-priority requirements implementations. The authentication system and user management system were finished.

Frontend The front-end team implemented marker and pipeline viewing and manipulation such as adding new markers/pipelines, updating the information, and deleting. We successfully visualized pipelines and markers on the leaflet-based map. The sidebar functionality was completed and now contained the information of the selected pipeline and/or marker. The profile page was developed as well.

Backend CRUD API endpoints for the sidebar information and profiles were designed and deployed on the backend. We also tested it using unit tests and ensured all API endpoints worked as expected.

E.5 Sprint 4 | Development

Mar 28 - Apr 10
In this sprint, we finished the left-over front-end part from the last sprint. And we also improved the coverage of the testing. We also started implementing nice-to-have functions based on the feedback from the client.

Frontend We added buttons for importing/exporting pipelines from/to a CSV file containing only coordinates. As requested by the client, the onsite survey can now be added to a marker. Furthermore, the distance between markers/pipeline points was able to be displayed. A button was added in the sidebar of a marker which can be used to redirect the user to Google Maps so that the user can be navigated to the marker.

Backend Corresponding CRUD API endpoints were deployed to the backend and tested. All the API endpoints were tested with high coverage of functions of the endpoints.

E.6 Sprint 5 | Finalization

Apr 11 - Apr 19
In the last sprint, we went through all the testing both for the front-end and back-end to ensure that the system is reliable. We fixed some bugs found during the testing.

As a team, we worked on the final presentation and finished the design report.