



TED UNIVERSITY

CMPE492 – Senior Project II

Search and Rescue Operation Portal(SAROP)

Final Report

Project Name: SAROP (Search and Rescue Operation Portal)

Project Url: sarop.tech

Team Members:

- Arda Gök (10016049214)
- Saliha Nursu Baltacı (48004738666)
- Ceren Özdoğan (13606132136)
- Mert Çıkla (33617203274)

Name of the supervisor: Emin Kuğu

Names of the juries: Tolga Kurtuluş Çapın - Venera Adanova

TABLE OF CONTENTS

1. Introduction.....	2
1.1 Description.....	3
1.2 Constraints.....	3
1.3 Professional and Ethical Issues	4
2. System Requirements	5
2.1 Features of Project	5
3. Architecture.....	7
3.1 Architecture in Backend.....	7
3.2 Architecture and design in Mobile.....	10
3.3Architecture and design on Frontend	14
4. Impact of the Project & Evaluation.....	31
5. Testing.....	33
6.Conclusion	35

1. Introduction

Identifying and executing a successful graduation project is a crucial milestone for the future careers of all senior students. With this in mind, we, a group of four friends united by our shared work ethics, principles, and vision, embarked on a journey to create a meaningful project that could make a significant impact. Our extensive discussions and research led us to a stark realization: the devastating earthquake that struck on February 6 exposed severe shortcomings in the organization and management of search and rescue operations, resulting in a tragic loss of life.

Motivated by this tragedy and the urgent need for better-coordinated emergency responses, we decided to develop the Search and Rescue Operation Portal (SAROP). SAROP is designed to address the inefficiencies observed during disaster response efforts, providing search and rescue teams with a powerful tool to manage their operations more effectively.

Our portal offers a secure authentication system, ensuring that only authorized team members can access sensitive information. Once logged in, users can enter new operations, update existing ones, and view current operations. Additionally, SAROP enables teams to drawing notes and polygons on maps and viewing them. It also allows users to switch between different maps. They can also see their location and search a specific location on map.

The backend of SAROP is developed using Java Spring Boot, ensuring a robust and scalable infrastructure. We utilize PostgreSQL for our database management, leveraging its powerful capabilities to store and retrieve the necessary data efficiently. The frontend is implemented using HTML, CSS, and JavaScript, providing a responsive and user-friendly interface that ensures ease of use for all team members. Additionally, we developed a mobile application using Flutter, enabling field users to access the system via their mobile devices, even in remote locations.

By integrating these features, SAROP aims to streamline the management of search and rescue operations, ultimately minimizing the damage caused by natural disasters and enhancing the efficiency of response teams. This project represents our collective effort to contribute to disaster management and serves as a testament to our commitment to leveraging technology for the greater good. We believe that SAROP will play a vital role in improving the coordination and effectiveness of search and rescue operations, potentially saving lives and reducing the impact of future catastrophes.

1.1 Description

The Search and Rescue Operation Portal (SAROP) is a comprehensive tool designed to enhance the efficiency and effectiveness of search and rescue teams. The portal's primary features include:

- **Authentication System:** Ensures that only authorized team members can access the portal, providing security and data privacy.
- **Operation Management:** Allows team members to enter new operations, update ongoing ones, and view current operations, facilitating better coordination and resource allocation.
- **Polygon and Tracking:** Enables teams to track different polygons on the map, aiding in the planning and execution of search and rescue missions.
- **User and Team Management:** Allows to manage the CRUD operations of users and teams of search and rescue teams
- **Mobile Application:** Developed using Flutter, the mobile application allows field users to access SAROP functionalities on their mobile devices, ensuring accessibility in remote areas.

The backend of SAROP is built using Java Spring Boot, and it interacts with a PostgreSQL database to manage and retrieve data efficiently. The frontend, created with HTML, CSS, and JavaScript, offers a user-friendly interface that facilitates easy navigation and usage.

1.2 Constraints

The development and implementation of SAROP are subject to several constraints:

- **Technical Constraints:** Ensuring compatibility across various devices and operating systems, along with reliable offline functionality.
- **Data Privacy:** Protecting sensitive information through robust encryption and secure authentication mechanisms.
- **Resource Allocation:** Efficient use of server capacity and database management to ensure system reliability and performance.
- **User Training:** Providing adequate training to search and rescue teams to ensure effective use of the portal.
- **Regulatory Compliance:** Adhering to relevant data protection and privacy laws to maintain legal and ethical standards.

1.3 Professional and Ethical Issues

Developing SAROP involves several professional and ethical considerations:

- **Data Security and Privacy:** Ensuring that sensitive information, such as operation details and personal data of team members, is securely stored and transmitted. Implementing strong encryption methods and secure authentication processes addresses these concerns.
- **Accuracy and Reliability:** Providing accurate and reliable information to avoid misleading search and rescue efforts. Rigorous testing and validation are essential to ensure data integrity.
- **User Accessibility:** Making the system accessible to all users, including those with disabilities, by implementing features that comply with accessibility standards.
- **Transparency and Accountability:** Maintaining transparent communication about the system's capabilities and limitations to manage user expectations, along with establishing accountability mechanisms to address any issues that arise during use.
- **Ethical Use of Technology:** Ensuring the portal is used ethically and responsibly, respecting user privacy, avoiding misuse of data, and ensuring the technology serves its intended purpose of improving disaster response efforts.

By addressing these professional and ethical issues, we aim to create a reliable, secure, and effective tool that significantly enhances the capabilities of search and rescue teams, ultimately contributing to saving lives and reducing the impact of natural disasters.

2. System Requirements

The software requirements for our project includes several essential components to ensure good functionality and efficient performance. On the backend, we have used Java 17 and Spring Boot 3 to build the core application. For database management, PostgreSQL version 14 or above is used, providing a reliable and scalable solution for data storage. Geoserver version 2.25 is used for processing and serving spatial data. Development on the backend is handled using IntelliJ IDEA. On the frontend, we have used HTML, CSS, and JavaScript to create an interactive user interface, integrating Leaflet for spatial data operations.

Additionally, the mobile application is developed using Flutter, with flutter_map used for map integration.

2.1 Features of Project

Our project have several features with different entities. You can see the features of the project at the below:

- **Authentication and Role Management**
 - Allowing users to register with their name, email and password.
 - Allowing users to login with their email and password.
 - Password of users is stored in the database by encoding the password.
 - The access of users is enabled with access token which is created using jwt tokens.
 - Enabling users to do several operations based on their roles.
- **User Management:**
 - Create, read, update, and delete (CRUD) operations for users.
 - Assigning each user to a specific team.
 - Assigning specific roles to each user such as Operation admin, admin and user
- **Team Management:**
 - CRUD operations for team entities.
 - Assign team members as users and assigning multiple locations to each team.
- **Category Management**
 - CRUD operations for category entities.
- **Team Location Management:**
 - CRUD operations for team location entities.

- **Operation Management:**
 - CRUD operations for operation entities.
 - Assigning a team to an operation
 - Assigning a category to an operation
 - Linking with multiple maps
- **Spatial Data Processing:**
 - Integrate Geoserver for serving spatial data.
 - Use Leaflet to visualize and interact with maps on the frontend.
 - Adding notes and polygons on the map using leaflet libraries.
 - Seeing the location of ourselves on the map.
 - Switching between the maps such as open street map and the map which is uploaded
- **Mobile Application:**
 - Authentication with email and password.
 - Integrate spatial data visualization and interaction using flutter_map.

3. Architecture

3.1 Architecture in Backend

The backend architecture of our project is organized into a modular structure to promote maintainability, scalability, and ease of development. The core of the backend is built using Java 17 and Spring Boot 3, following a clean and well-organized package structure.

1. **Root Package: com.sarop.saropbackend**

- This is the base package containing all the sub-packages and classes necessary for the application.

2. **Sub-packages:**

- **authentication:** Handles authentication logic, including login, registration, and token management.
- **category:** Manages the CRUD operations related to categories.
- **common:** Contains common utility classes and shared resources used across various modules.
- **config:** Contains configuration classes for setting up application properties, security configurations, and other Spring Boot configurations.
- **note:** Handles operations related to notes on specific locations on map.
- **operation:** Manages CRUD operations of operations as well as linking them with categories, teams, and spatial data.
- **polygon:** Manages spatial data and operations related to polygons, likely representing areas on a map.
- **restapi:** Provides REST API endpoints for external communication with geoserver
- **team:** Manages team-related operations, including CRUD operations and team management functionalities.
- **teamLocation:** Manages CRUD operations for team locations.

3. Inside of packages

- **controller:** Contains controllers which handle HTTP requests related to operations for the specific entity.
- **dto:** Contains Data Transfer Objects (DTOs) used for transferring data between the client and server. This includes request and response objects.
- **model:** Defines the models which represent the database entities.
- **repository:** Contains repository, which extends Spring Data JPA repositories to perform CRUD operations on user entities.
- **service:** Contains the service layer, including interfaces and implementations which encapsulate the business logic related to models.

Besides we have several relations between the entities in our project.

- **User and Team:**
 - A User belongs to one Team (ManyToOne relationship).
 - A Team has one User as the teamLeader (OneToOne relationship).
 - A Team has many Users as members (OneToMany relationship).
- **Team and TeamLocation:**
 - A TeamLocation belongs to one Team (ManyToOne relationship).
 - A Team has many TeamLocations (OneToMany relationship).
- **Team and Operation:**
 - An Operation belongs to one Team (ManyToOne relationship).
 - A Team has many Operations (OneToMany relationship).
- **Operation and Category:**
 - An Operation belongs to one Category (ManyToOne relationship).
 - A Category has many Operations (OneToMany relationship).

- **Operation and Map:**
 - An Operation can have many Maps (ManyToMany relationship with join table).
 - A Map can be linked to many Operations (ManyToMany relationship with join table).

- **Map and Workspace:**
 - A Map belongs to one Workspace (ManyToOne relationship).
 - A Workspace has many Maps (OneToMany relationship).

- **Map and Note:**
 - A Note belongs to one Map (ManyToOne relationship).
 - A Map has many Notes (OneToMany relationship).

- **Map and Polygon:**
 - A Polygon belongs to one Map (ManyToOne relationship).
 - A Map has many Polygons (OneToMany relationship).

- **Note and User:**
 - A Note belongs to one User (ManyToOne relationship).
 - A User has many Notes (OneToMany relationship).

- **Note and Coordinate:**
 - A Note has one Coordinate (OneToOne relationship).

- **Polygon and Coordinate:**
 - A Polygon has many Coordinates (OneToMany relationship).

3.2 Architecture and design in Mobile

General Architecture of Mobile Application

We are utilizing Flutter to construct our mobile app. Google created the Flutter UI toolkit, which enables developers to create native apps for the iOS and Android operating systems. According to customer's request in mobile application users just getting the maps, notes, polygons from backend and geoserver. There is no update, post process in the mobile application except register and login. Our program is divided into three primary modules:

Authentication Module: Includes pages for registration and login where users can sign in or make new accounts. Our API is used in interaction with this module.

Workspace Module: Users can choose from a variety of workplaces and see the maps associated with them in this module. Workspace and map viewing is geoserver based. The flutter application and geoserver are mediated by the API.

Map Module: This module allows user to see the maps of the workspace that user has selected, toggling between Google Maps and the presented map, and displaying notes and polygons on the screen. This uses geoserver-based processing for map presentation. The flutter application and geoserver are mediated using an API.

API Integration

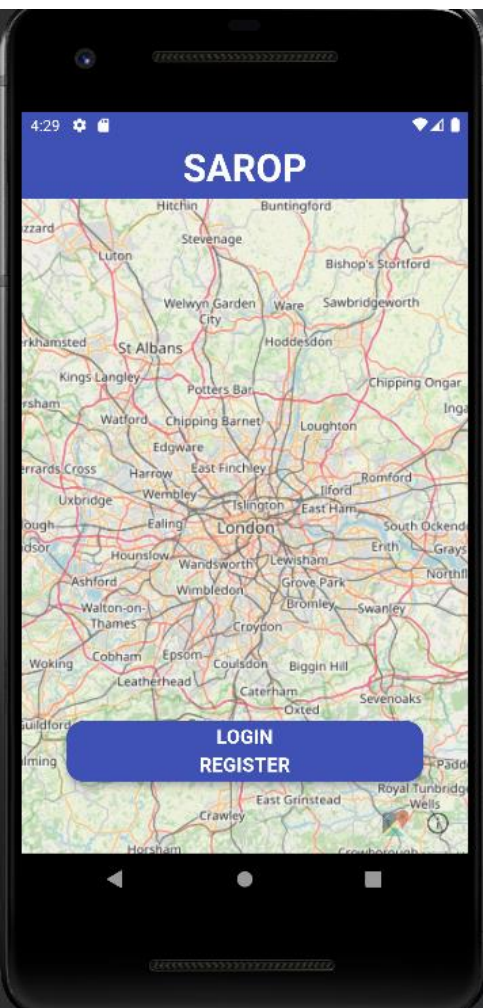
Our mobile application uses a RESTful API to interface with the backend, Spring Boot. Secure HTTPS transmission of data, including workspace and layer selections, user login and registration details, is made to the backend. The responses are sent back to the mobile application by the backend once the required tasks have been completed. The data security and application reliability are enhanced by this arrangement.

Map Integration

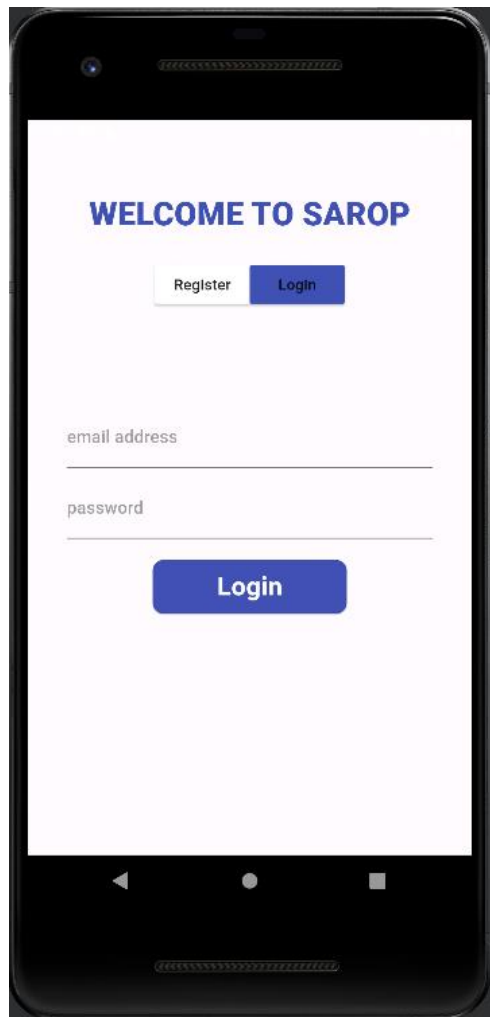
Our program integrates GeoServer with its map functions. A platform called GeoServer is used to maintain and display maps and geographic data. Google Maps and ECW format maps are dynamically loaded and shown based on user requirements. GeoServer and the backend are used to record and handle operations like notes and polygons drawn on the map.

Design of Mobile Application

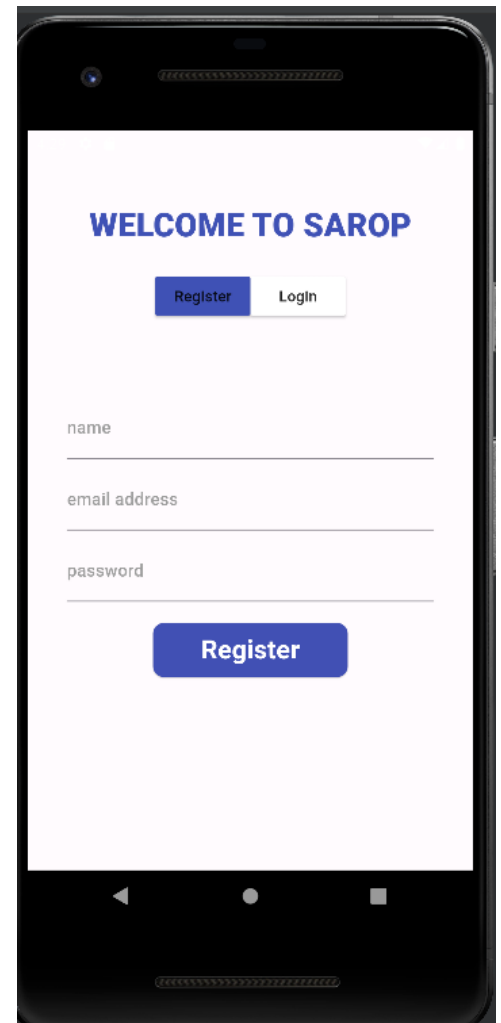
1. Welcoming-Login-Register Pages



Page 1



Page 3

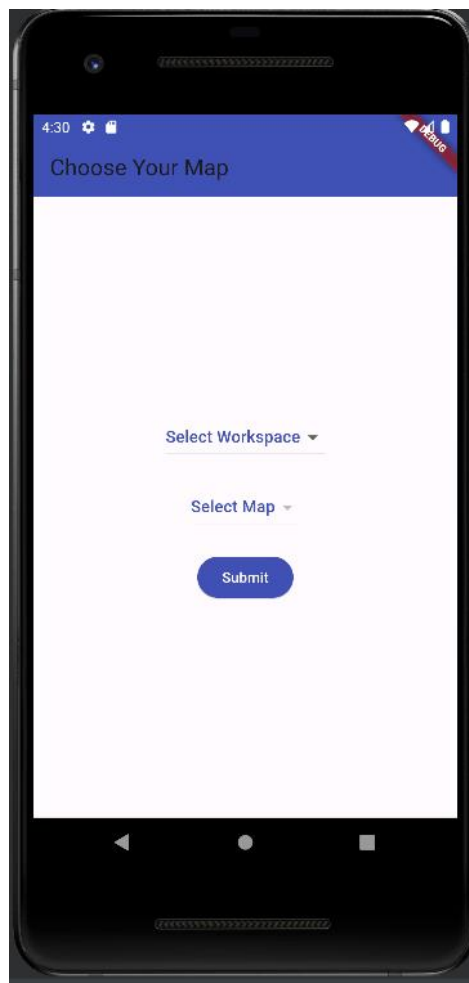


Page 2

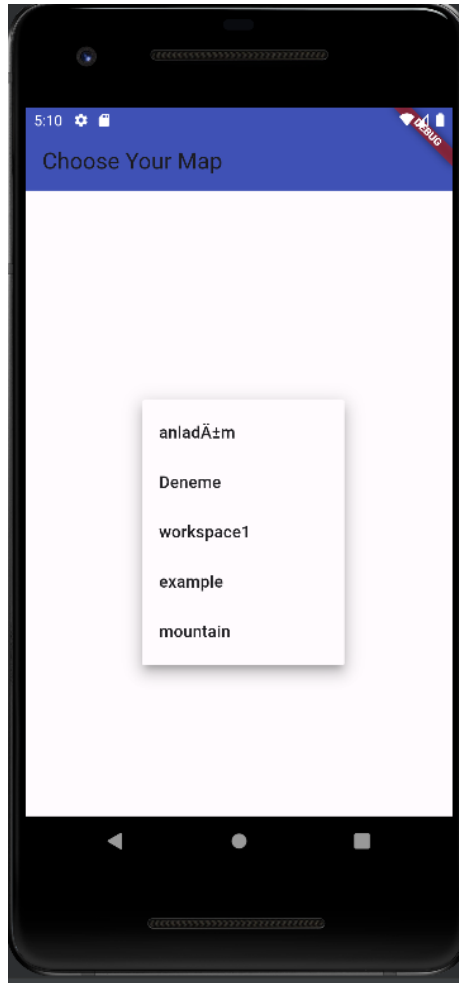
The welcome page is the first screen we encounter when we launch our mobile application. A map that was retrieved from a map server is visible behind this screen. We included this function in order to comply with the Apple Store's mandate that certain features must be accessible without logging in.

We can go between the two pages of this screen by pressing the login or register button, which brings up a registration/login screen. Users have the option to register on this screen. After the admin gives their approval, their registration data is transmitted to the backend, and they may use authentication to log in via the login screen. The API serves as the link between all of these activities.

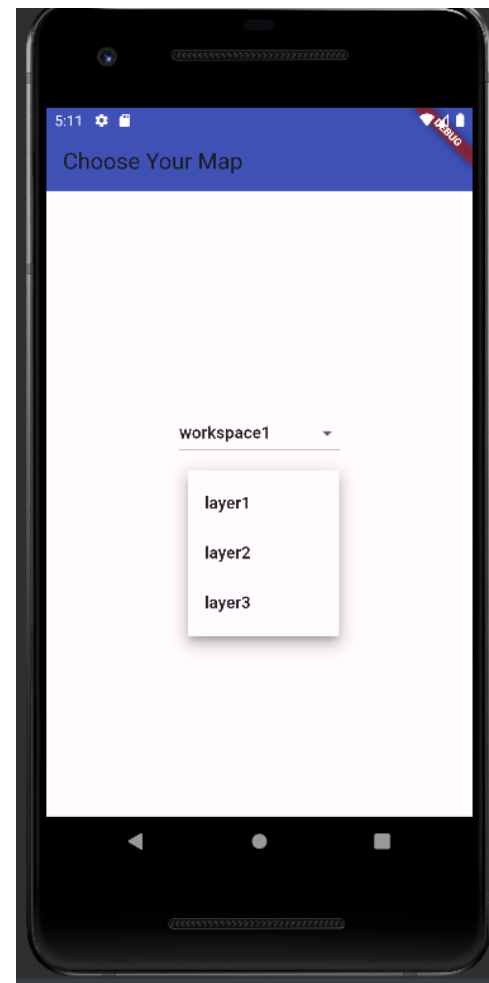
2. Choose Your Map Page



Page 6



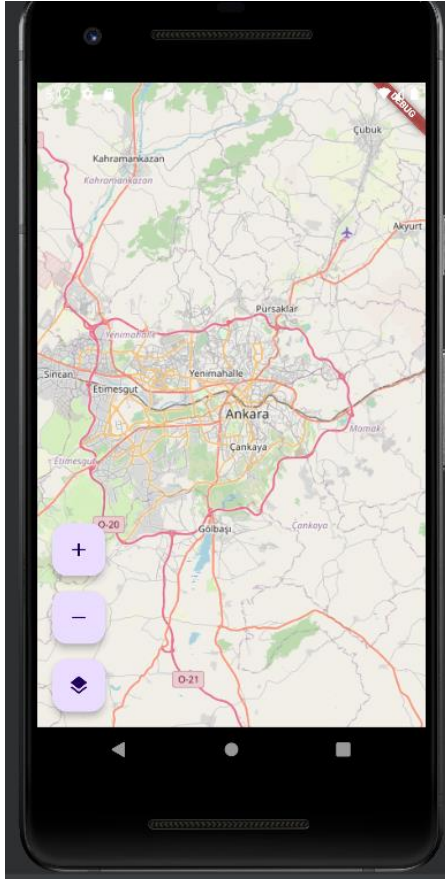
Page 5



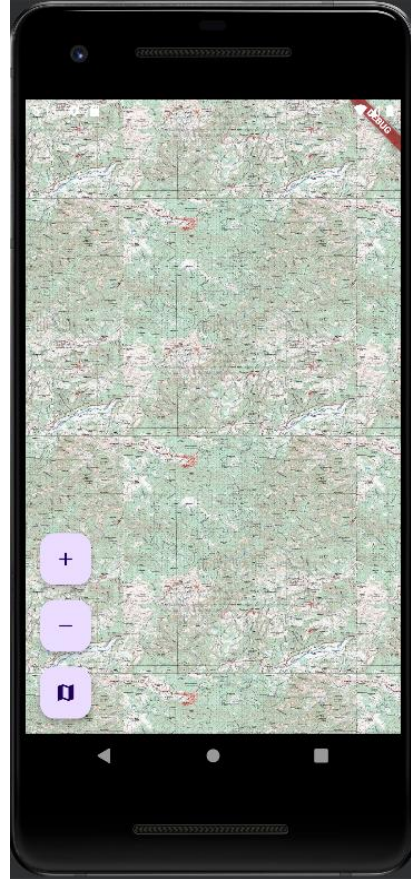
Page 4

Users can examine the registered workspaces in the database and choose a map from the appropriate workspace on this page. Through REST API, all of these processes are coordinated between GeoServer and the backend. Users are redirected to a new screen upon picking a workspace and its associated map and pressing the submit button.

3. Get Your Map Page



Page 7



Page 8

The map that was chosen and uploaded on the "choose your map" page is retrieved from GeoServer via a REST API and shown in the mobile application using FlutterMap on this new screen. This screen also shows notes that are currently open and polygons that the administrator has drawn on our website during this procedure. There is also an option to switch to OpenStreetMap on this screen. Users have the option to always remain on the ECW map or to always switch to OpenStreetMap.

3.3 Architecture and design on Frontend

The frontend of our project was developed using HTML, CSS, and JavaScript, with the additional integration of Bootstrap 5 to ensure a responsive and visually appealing user interface.

Technologies Used

- **HTML, CSS, JavaScript:** The core technologies used to structure, style, and add interactivity to the web pages.
- **Bootstrap 5:** Utilized for its extensive prebuilt components and responsive grid system, which helped in creating a modern and consistent design across the application.
- **Leaflet.js:** Used for advanced mapping functionalities, providing an interactive map interface with extensive customization.

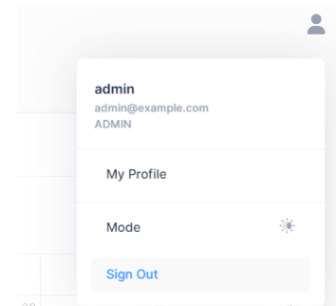
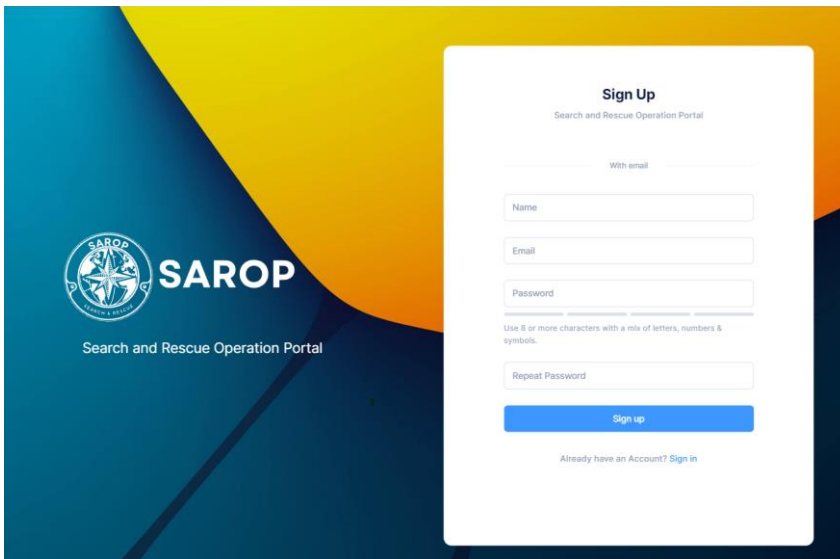
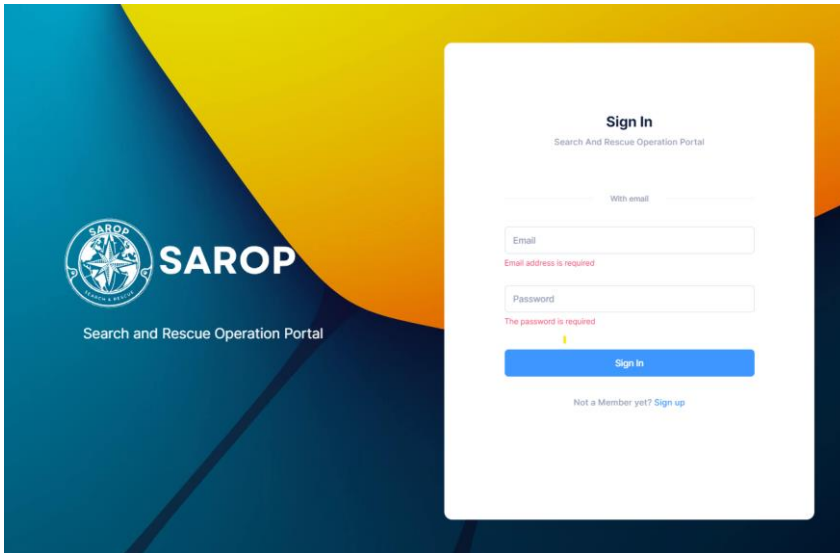
Functionalities Implemented

- **AJAX Operations:** JavaScript was employed to handle asynchronous HTTP requests using GET, POST, and DELETE methods. These operations enabled seamless interaction with the backend, allowing for the retrieval, submission, and deletion of data without requiring page reloads.
- **User and Admin Views:** The frontend dynamically adjusts the visible content based on the user type (admin or standard user). While the admin has access to all pages, standard users have restricted access, excluding admin-specific functionalities.
 - **Admin-Specific Functionalities:**
 - **Category List:** Allows admins to manage categories within the application.
 - **Team Location List:** Enables admins to view and manage the locations of various teams.
 - **Team List:** Provides admins with an overview and management options for different teams.
 - **User List:** Facilitates the management of user accounts by the admin.

- **Shared Functionalities:**
 - **Profile Details:** Users and admin can view their profile information.
 - **Map:** Operation admins and admin can add operations to the calendar based on the map data. Users, operation admins and admins can add notes and polygons as well as viewing and deleting them.
 - **Calendar:** Operation admins and admin can add operations to the calendar, linked to specific locations on the map, facilitating organized and efficient operation planning.

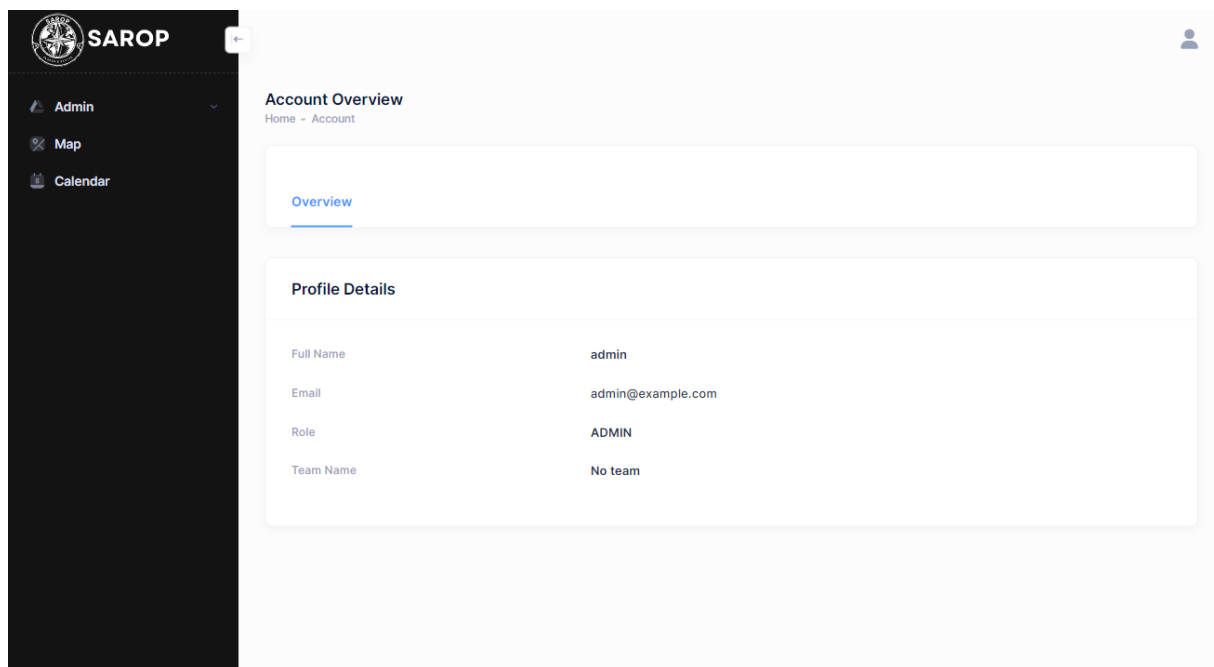
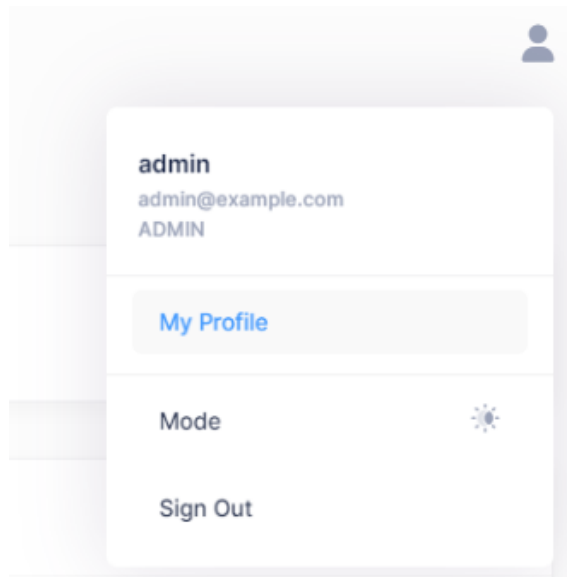
User Authentication

- **Sign In/Sign Out/Sign Up Pages:** These pages handle user authentication, allowing users to create accounts, sign in, and sign out securely.



My Profile Page

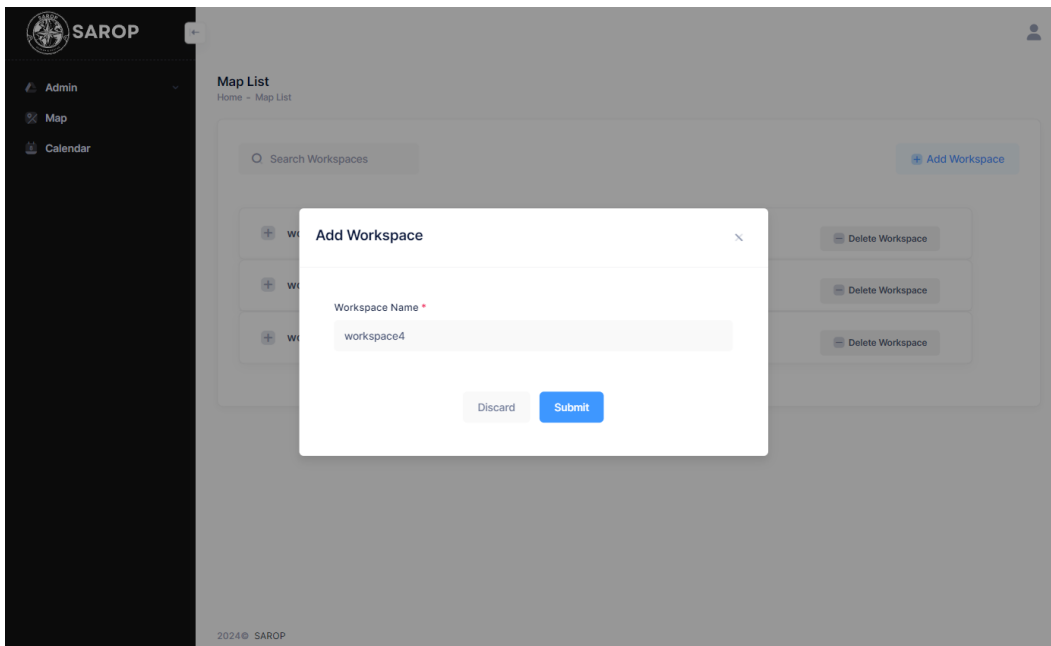
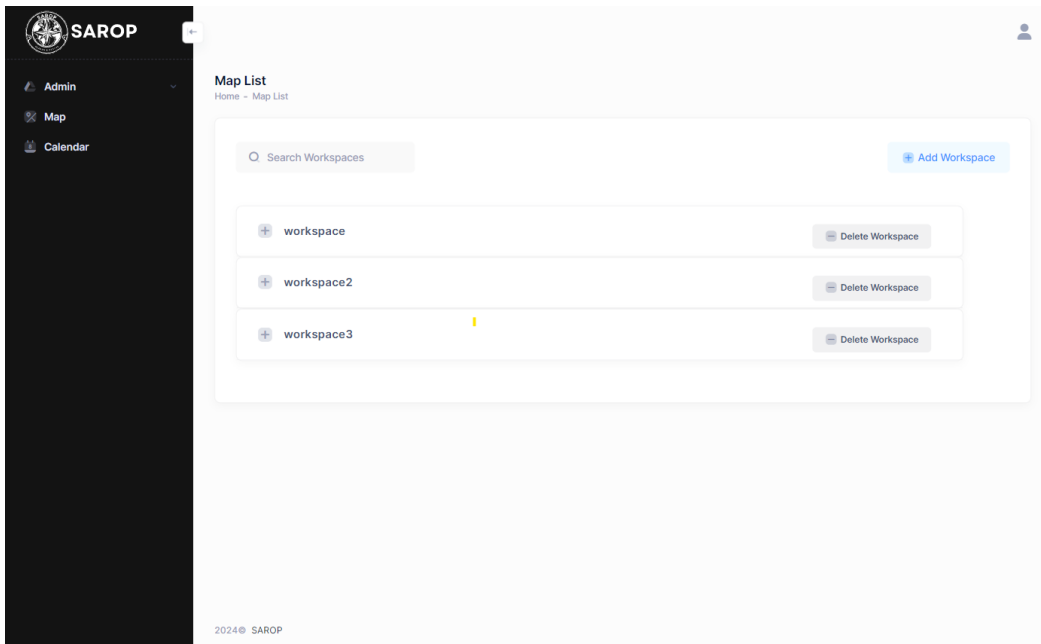
- The My Profile page allows users to view and update their personal information. The details displayed on **the** profile page include:
 - Full Name
 - Email
 - Role
 - Team Name



Workspace and Layer Management

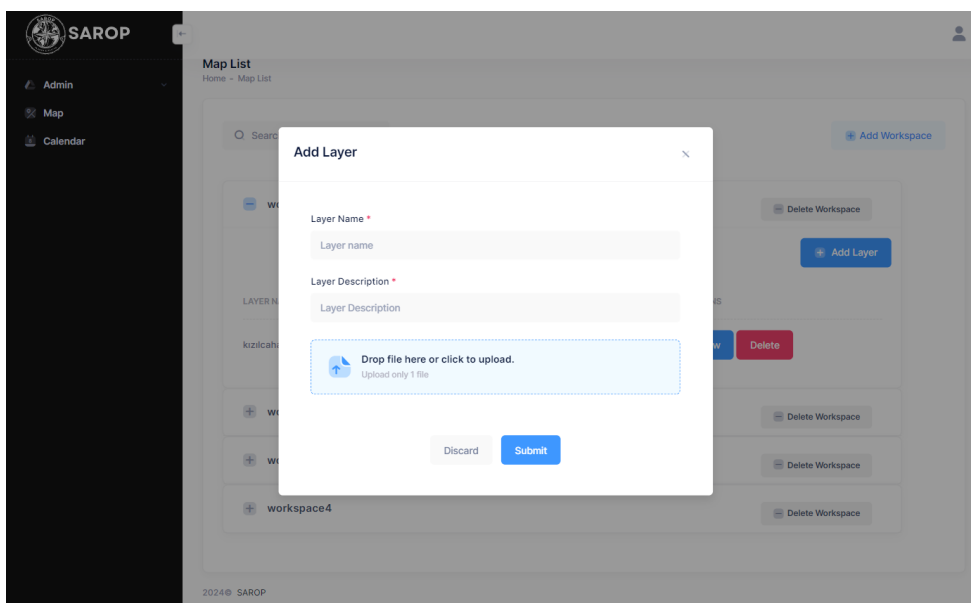
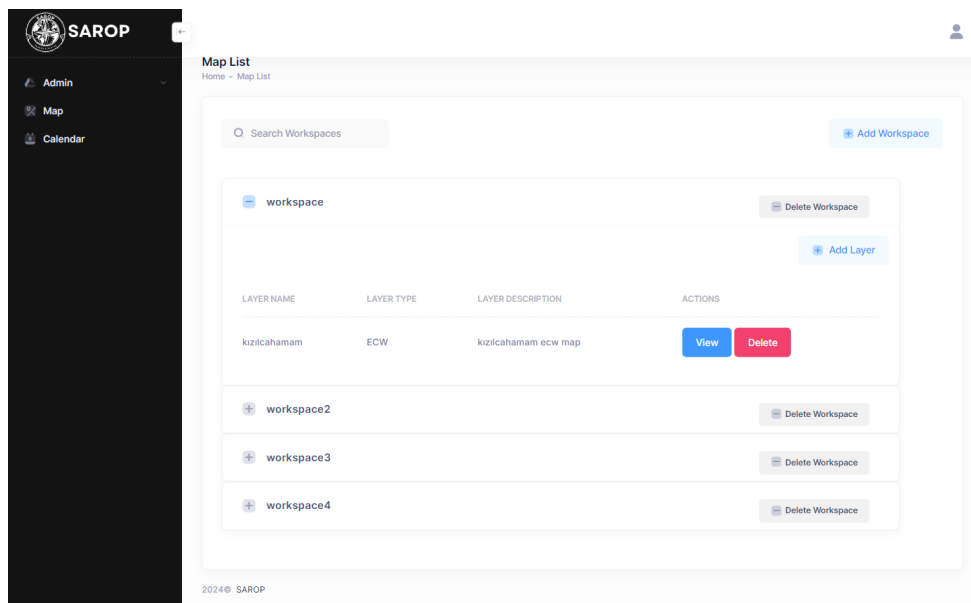
Before accessing the map page, users can interact with the **Map List** page where they can manage their workspaces and layers:

- **Workspace Management:** Users can view existing workspaces, add new workspaces by required workspace name input, and delete existing ones.
 - **Add Workspace:** Users can add a new workspace by clicking the "Add Workspace" button and providing a Workspace Name (required).



- **Layer Management:** Within each workspace, users can view the details of layers, including layer name, layer type, and layer description. They can also add new layers, delete layers, and use the view button to navigate to the map page with the selected layer and workspace.
 - **Add Layer:** Users can add a new layer by clicking the "Add Layer" button and providing the following required information:
 - Layer Name (required)
 - Layer Description (required)

- **File Drop (required):** Allows uploading one file of types tif, tiff, ecw, jpg, jpeg only.



Map Integration

- **Leaflet.js:** For the mapping functionalities, Leaflet.js was integrated and customized to fit our specific requirements. This included adapting functionalities such as note adding and polygon drawing (including measurements) to integrate seamlessly with our UI, allowing users to:
 - **Zoom In/Out:** Easily adjust the map view.
 - **Fullscreen Mode:** Switch to a full screen map view.

- **Printing Options:** Print the map with various options such as portrait, landscape, auto, and custom settings.
- **Layer Control:** Toggle between different map layers.
- **Scale Display:** View the scale of distances on the map.
- **Note Markers:** Add, remove, and toggle visibility of note markers, customized to fit our specific UI requirements.
- **Live Location Tracking:** Show the live location of the logged-in user.
- **Polygon Tools:** Measure distances and areas on the map, with options to save or delete these measurements as polygon.

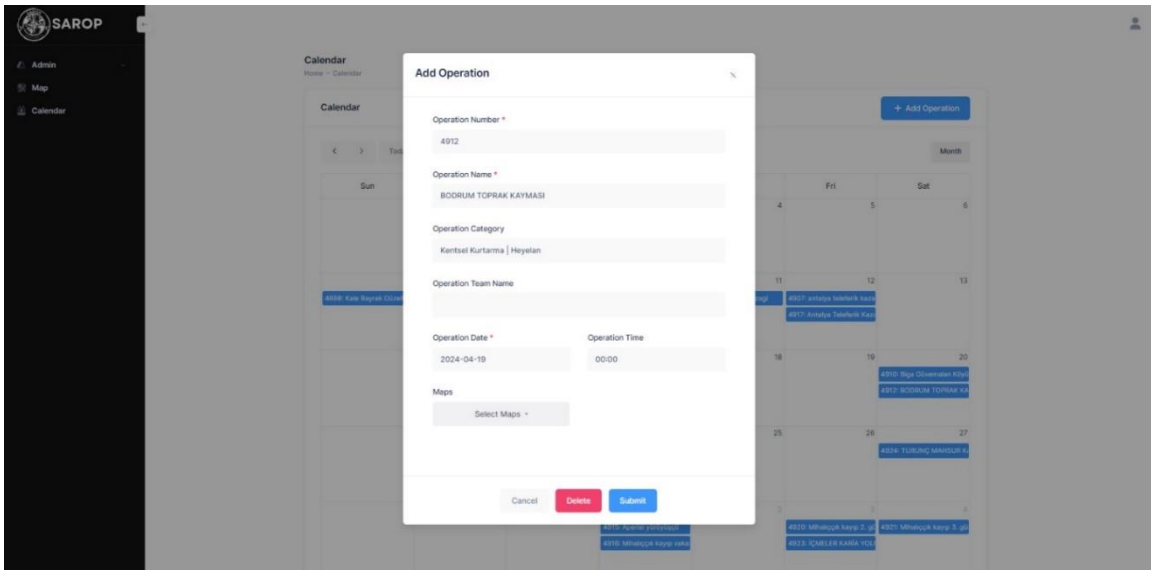
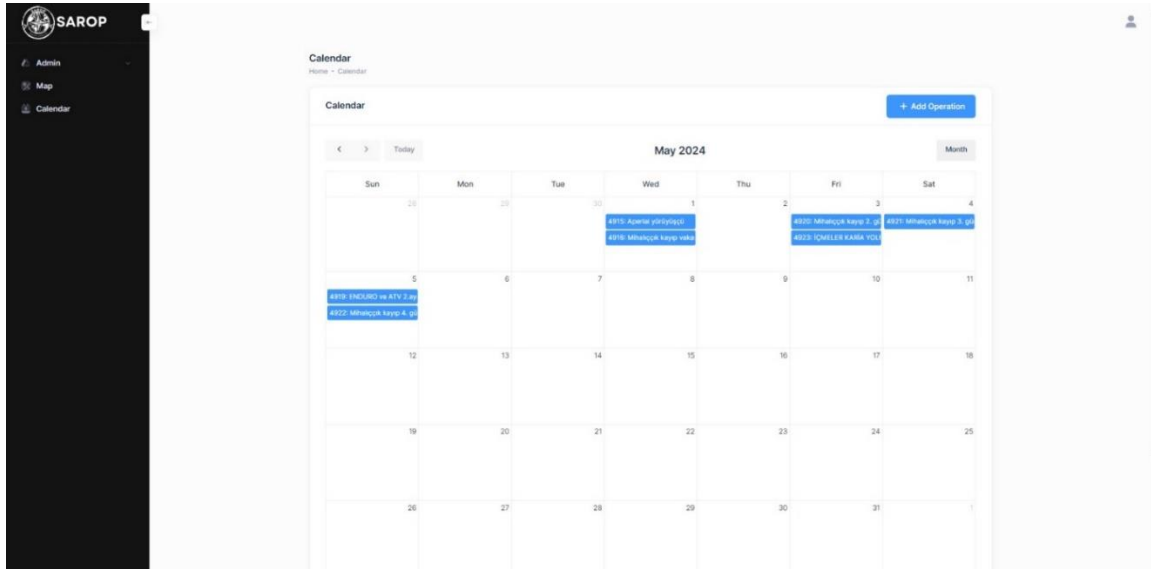
These tools were adapted for our custom UI.



Calendar Integration

- **Operation Scheduling:** Users and admins can add, delete and view operations to the calendar, facilitating organized and efficient operation planning. The add operation button include:
 - Operation Number (required)
 - Operation Name (required)
 - Operation Category

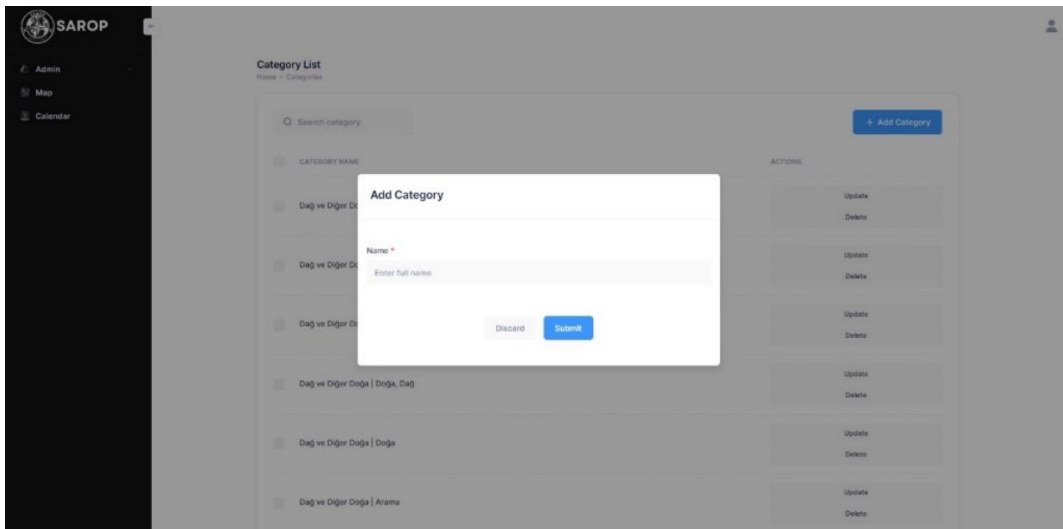
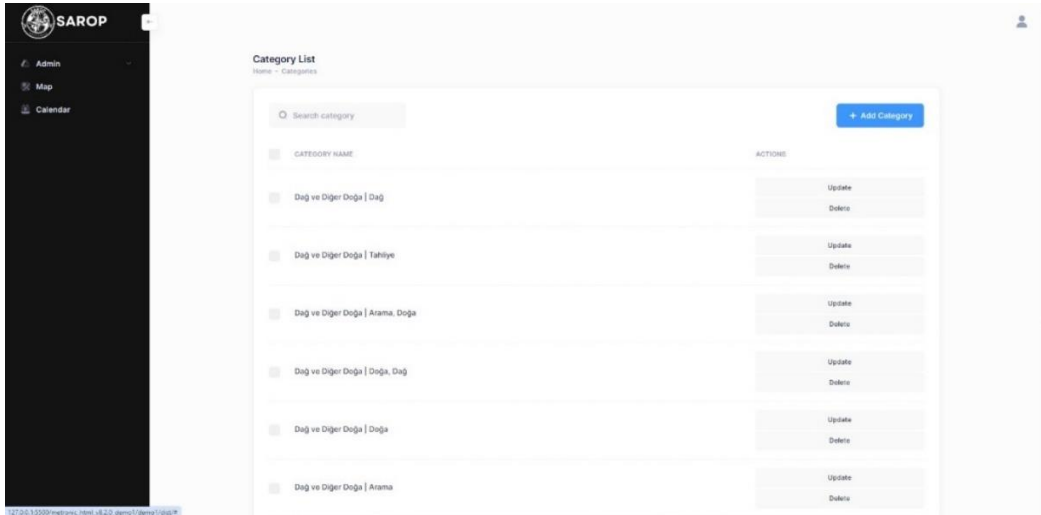
- Operation Team Name
- Operation Date (required)
- Operation Time
- Map Selection

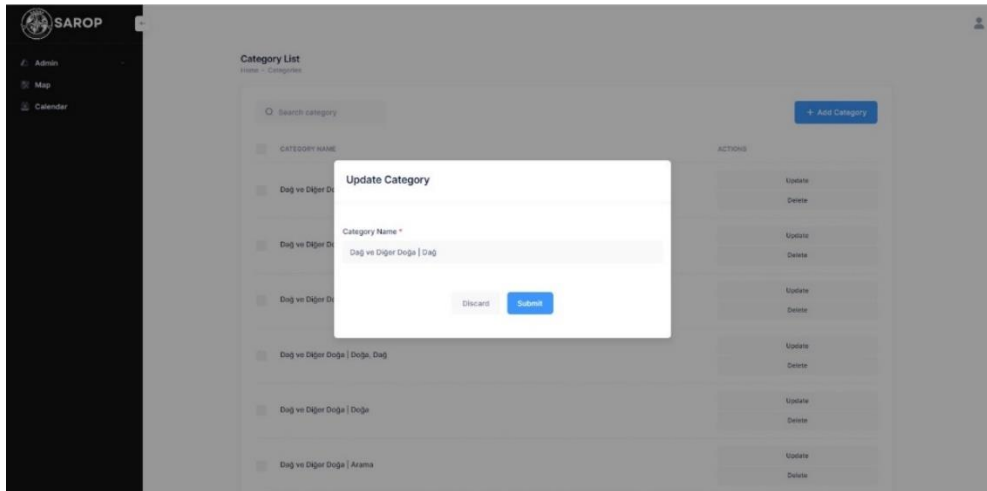


Admin Pages:

- **Category List:** Allows admin to view the details of categories, including category name. Admin can also add new categories, delete categories, and use the update button to update selected category.
 - **Add Category:** Admin can add a new category by clicking the "Add Category" button and providing the following required information:

- Category Name (required)
- **Update Category:** Admin can update existing category by clicking the "Update" button and providing the following required information:
 - Category Name (required)

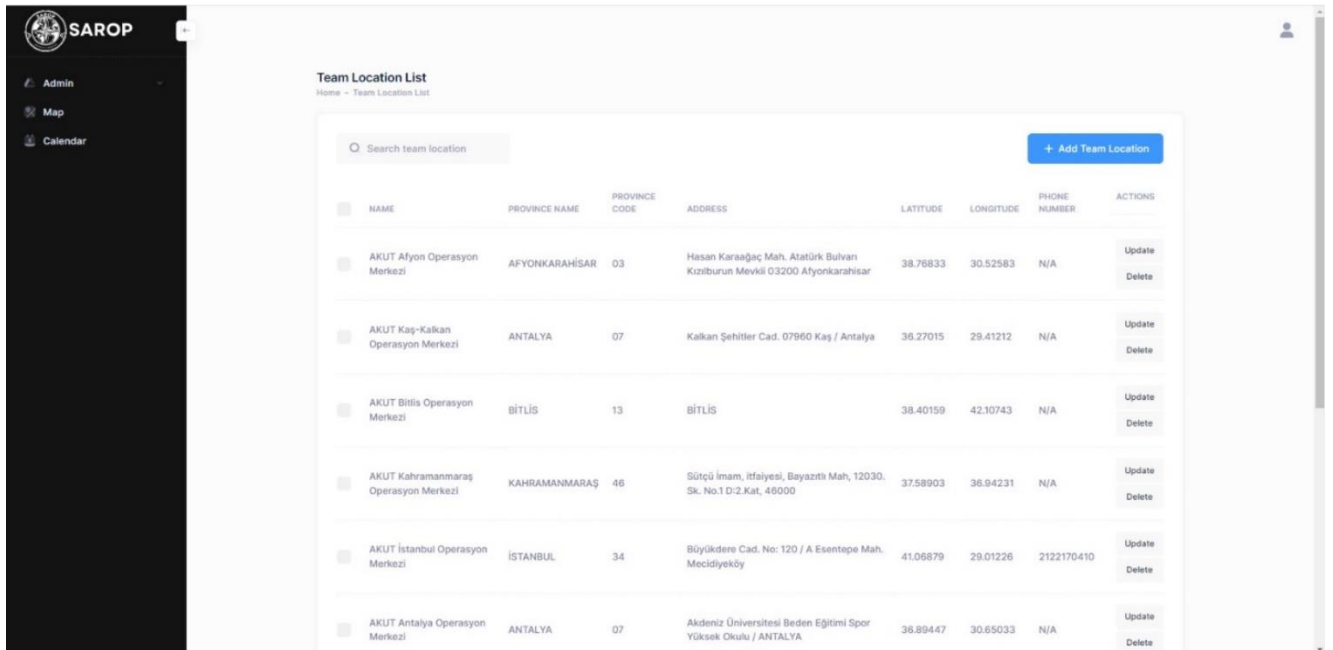




- **Team Location List:** Allows admin to view the details of team locations, including name, province name, province code, addresses, latitude, longitude, and phone number. Admin can also add new team location, delete team locations, and use the update button to update selected team location.
 - **Add Team Location:** Admin can add a new team location by clicking the "Add Team Location" button and providing the following required information:
 - Name (required)
 - Province Name (required)
 - Province Code (required)
 - Address (required)
 - Latitude (required)
 - Longitude (required)
 - Description
 - Phone Number (required)
 - Second Phone Number
 - Third Phone Number
 - Fax Number

- **Update Team Location:** Admin can update existing team location by clicking the "Update" button and providing the following required information:


- Name (required)
- Province Name (required)
- Province Code (required)
- Address (required)
- Latitude (required)
- Longitude (required)
- Description
- Phone Number
- Second Phone Number
- Third Phone Number
- Fax Number



Team Location List
Home - Team Location List

Search team location [+ Add Team Location](#)

NAME	PROVINCE NAME	PROVINCE CODE	ADDRESS	LATITUDE	LONGITUDE	PHONE NUMBER	ACTIONS
AKUT Afyon Operasyon Merkezi	AFYONKARAHISAR	03	Hasan Karaağaç Mah. Atatürk Bulvarı Kızıburun Mevkii 03200 Afyonkarahisar	38.76833	30.52583	N/A	Update Delete
AKUT Kaş-Kalkan Operasyon Merkezi	ANTALYA	07	Kalkan Şehitler Cad. 07960 Kaş / Antalya	36.27015	29.41212	N/A	Update Delete
AKUT Bitlis Operasyon Merkezi	BİTLİS	13	BİTLİS	38.40159	42.10743	N/A	Update Delete
AKUT Kahramanmaraş Operasyon Merkezi	KAHRAMANMARAŞ	46	Sütçü İmam, İtfaiyesi, Bayazıtı Mah, 12030. Sk. No:1 D:2.Kat, 46000	37.58903	36.94231	N/A	Update Delete
AKUT İstanbul Operasyon Merkezi	İSTANBUL	34	Büyükdere Cad. No: 120 / A Esentepe Mah. Mecidiyeköy	41.06879	29.01226	2122170410	Update Delete
AKUT Antalya Operasyon Merkezi	ANTALYA	07	Akdeniz Üniversitesi Beden Eğitimi Spor Yüksek Okulu / ANTALYA	36.80447	30.65033	N/A	Update Delete



- Admin
- Map
- Calendar

Team Location List

Home - Team Location List

Search team location

NAME	LATITUDE	LONGITUDE	PHONE NUMBER	ACTIONS
AKUT Afyon O Merkezi	38.76833	30.52583	N/A	Update Delete
AKUT Kaç-Kaç Operasyon Me	38.27015	29.41212	N/A	Update Delete
AKUT Bitlis Op Merkezi	38.40159	42.10743	N/A	Update Delete
AKUT Kahram Operasyon Me	37.58903	36.94231	N/A	Update Delete
AKUT İstanbul Merkezi	41.06879	29.01226	2122170410	Update Delete
AKUT Antalya Merkezi	36.89447	30.65033	N/A	Update Delete

+ Add Team Location

Add Team Location

Name: Enter full name

Province Name: Enter province name

Province Code: Enter province code

Address: Enter address


Latitude: Enter Latitude

Longitude: Enter Longitude

Description: Enter Description

Phone Number: Enter Phone Number

Second Phone Number:



- Admin
- Map
- Calendar

Team Location List

Home - Team Location List

Search team location

NAME	LATITUDE	LONGITUDE	PHONE NUMBER	ACTIONS
AKUT Afyon O Merkezi	38.76833	30.52583	N/A	Update Delete
AKUT Kaç-Kaç Operasyon Me	36.27015	29.41212	N/A	Update Delete
AKUT Bitlis Op Merkezi	38.40159	42.10743	N/A	Update Delete
AKUT Kahram Operasyon Me	37.58903	36.94231	N/A	Update Delete
AKUT İstanbul Merkezi	41.06879	29.01226	2122170410	Update Delete
AKUT Antalya Merkezi	36.89447	30.65033	N/A	Update Delete

+ Add Team Location

Update Team Location

Name * AKUT Afyon Operasyon Merkezi

Province Code * 03

Province Name * AFYONKARAHISAR

County Name * null

Address * Hasan Karaağaç Mah. Atatürk Bulvarı Kızılburun Mevkii 03200 Afyonkarahisar


Latitude * 38,76833

Longitude * 30,52583

Description null

Phone Number

- **Team List:** Allows admin view the details of team, including name, foundation year, province code, province name, team leader email, phone description. Admin can also add new team, delete team, and use the update button to update selected team.
 - **Add Team:** Admin can add a new team by clicking the "Add Team" button and providing the following required information:
 - Team Name (required)
 - Foundation Year (required)
 - Province Code (required)
 - Province Name (required)
 - Team Leader
 - Phone Description (required)
 - Team Locations
 - Users
 - **Update Team:** Admin can update existing team by clicking the "Update" button and providing the following required information:
 - Team Name (required)
 - Foundation Year (required)
 - Province Code (required)
 - Province Name (required)
 - Team Leader
 - Phone Description (required)
 - Team Locations
 - Users

**SAROP**


[Admin](#)
[Map](#)
[Calendar](#)

Team List

Home - Team List

+ Add Team

NAME	FOUNDATION YEAR	PROVINCE CODE	PROVINCE NAME	TEAM LEADER EMAIL	PHONE DESCRIPTION	ACTIONS
Antalya	1999	07	ANTALYA	yilmazsevgu@akut.org.tr	0532 256 44 37	Update Delete
Ankara	2000	06	ANKARA	gunalmahiyatagan@akut.org.tr	0 312 220 31 11 (İhbarlar için) (Diğer konular için) 0533 565 47 74	Update Delete
Marmaris	2001	48	MUĞLA	tolgagozum@akut.org.tr	0532 261 90 23	Update Delete
Bingöl	2002	12	BİNGÖL	ahmetates@akut.org.tr	0532 417 46 89	Update Delete
Kocaeli	2003	41	KOCAELİ	akutkocaeli@akut.org.tr	0530 879 6632 (İhbarlar için) (Diğer konular için) 0505 247 6406	Update Delete
Niğde	2005	51	NİĞDE	nedimurcan@akut.org.tr	0533 551 82 43	Update Delete

**SAROP**

[Admin](#)
[Map](#)
[Calendar](#)

Team List

Home - Team List

+ Add Team

Add Team

Team Name *

Foundation Year *

Province Code *
 Lütfen bu alanı doldurun.

Province Name *

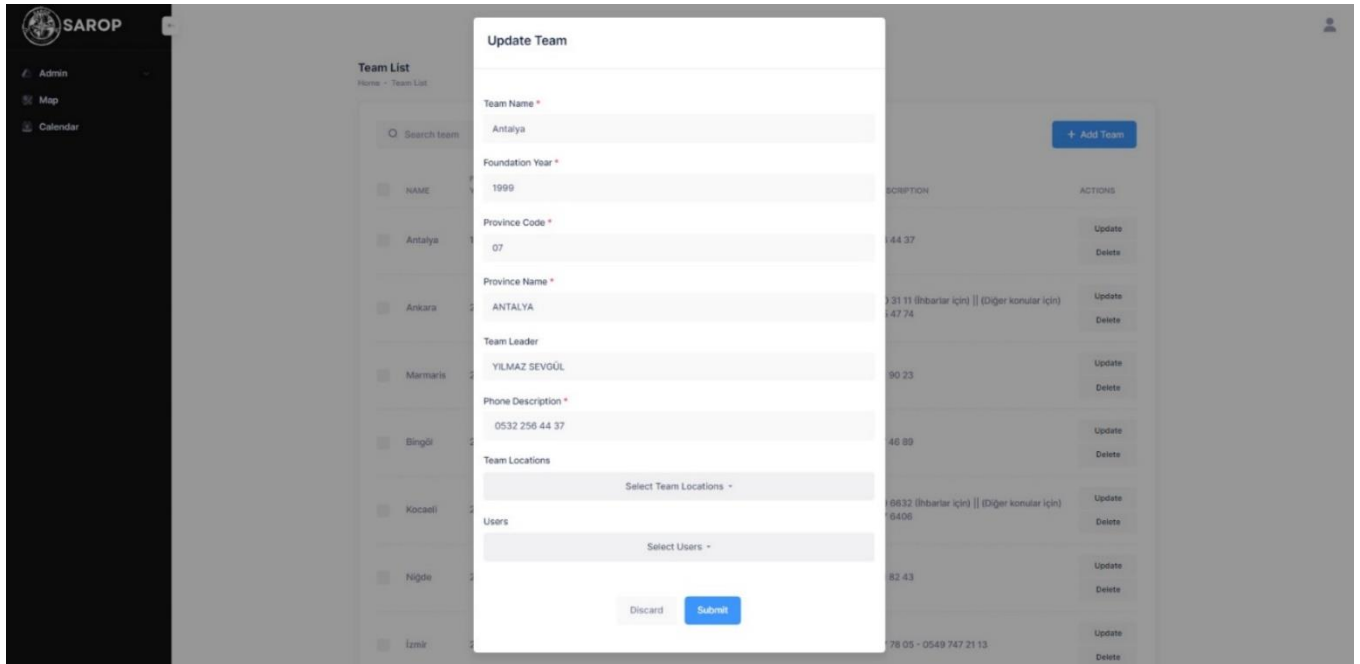
Team Leader

Phone Description *

Team Locations

Users

[Discard](#) [Submit](#)



- **User List:** Allows admin to view the details of users, including name, role, status, team name. Admin can also add new user, delete user, and use the update button to update selected user. Also use the “NonVerified List” button to navigate to the NonVerified List page.
 - **Add Team:** Admin can add a new user by clicking the "Add User" button and providing the following required information:
 - Full Name (required)
 - Email (required)
 - Password (required)
 - Role (required)
 - Team
 - **Update Team:** Admin can update existing user by clicking the "Update" button and providing the following required information:
 - Full Name (required)
 - Email (required)
 - Password (required)
 - Role (required)
 - Team

SAROP

- Admin
- Map
- Calendar

Users List
Home - Users List

NonVerified List + Add User

NAME	ROLE	STATUS	TEAM NAME	ACTIONS
EMİNE EDA KUŞÇU edakuscu@akut.org.tr	OPERATION_ADMIN	Enabled	Kahramanmaraş	Update Delete
admin admin@example.com	ADMIN	Enabled	Undefined	Update Delete
MUSTAFA TUNC TUNCER tuncuncer@akut.org.tr	OPERATION_ADMIN	Enabled	Undefined	Update Delete
YILMAZ SEVGİLİ yilmazsevgili@akut.org.tr	OPERATION_ADMIN	Enabled	Deneme2222	Update Delete
TOLGA GÖZLÜM tolgagolum@akut.org.tr	OPERATION_ADMIN	Enabled	Deneme2222	Update Delete
RAMAZAN ERGUT ramazanergut@akut.org.tr	OPERATION_ADMIN	Enabled	Deneme2222	Update Delete
Tolga Özdoğan ceren@example.com	USER	Enabled	Ankara	Update Delete

SAROP

- Admin
- Map
- Calendar

Users List
Home - Users List

NonVerified List + Add User

Add User

Full Name *
Full name

Email *
admin@example.com

Password *

Role *
 ADMIN
 OPERATION_ADMIN
 USER

Team
Antalya

Discard Submit

SAROP

- Admin
- Map
- Calendar

Users List
Home - Users List

NonVerified List + Add User

Update User

Name *
EMİNE EDA KUŞÇU

Email *
edakuscu@akut.org.tr

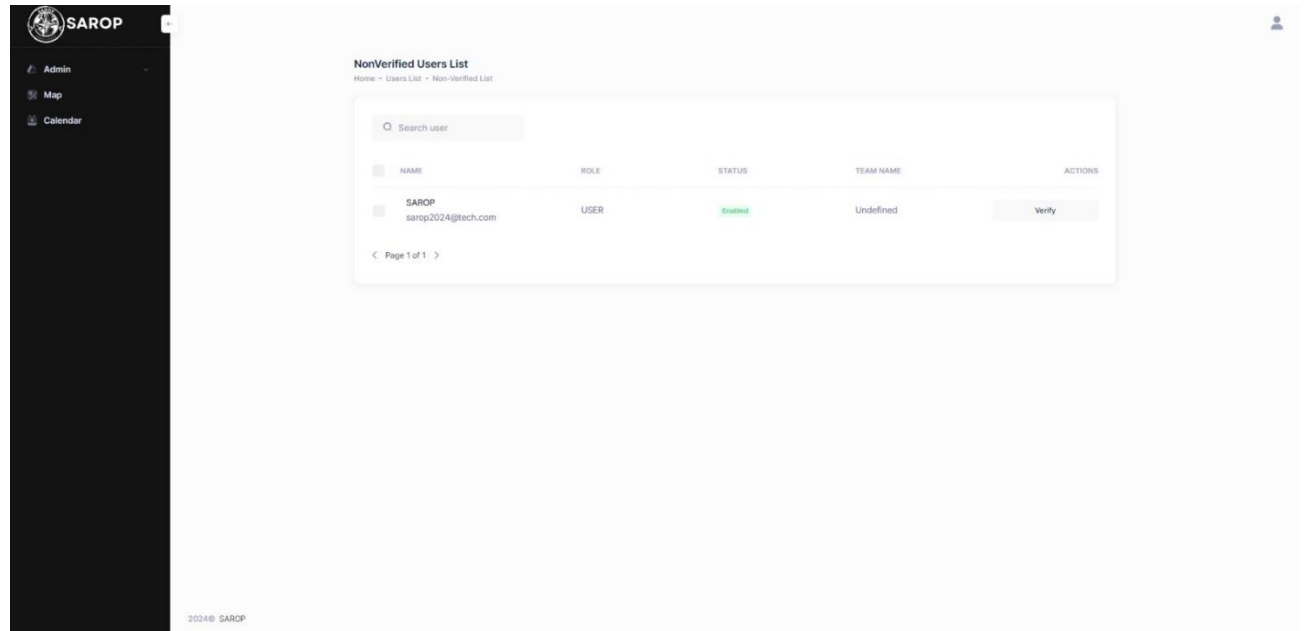
Password *

Role *
Operation Admin

Team
Kahramanmaraş

Discard Submit

- **NonVerified Users List:** Allows admin to view the nonverified list of users, including name, role, status, team name. Admin can also verify the user by clicking “Verify” button. In this way, the verified user is deleted from the nonverified list and the users list is passed.



Overall, the frontend architecture was designed to be robust, interactive, and user-friendly, catering to the needs of both regular users and admins, and significantly aiding in search and rescue operations through advanced mapping and scheduling functionalities.

4. Impact of the Project & Evaluation

The Search and Rescue Operation Portal (SAROP) has the potential to significantly improve the efficiency and effectiveness of search and rescue operations, especially in disaster-prone areas like Turkey. The key impacts of the project include:

- **Enhanced Coordination:** By providing a centralized platform for managing operations, SAROP enhances the coordination among search and rescue teams. This leads to more organized and effective responses to disasters.
- **Improved Resource Allocation:** The ability to track paths and allocate teams to specific regions ensures that resources are used optimally, increasing the chances of successful rescue missions.
- **Data Security and Privacy:** With robust authentication and encryption mechanisms, SAROP ensures that sensitive information is protected, maintaining the privacy and security of the data.
- **User-Friendly Interface:** The implementation using HTML, CSS, and JavaScript provides a responsive and intuitive interface, making it easy for team members to use the portal effectively.
- **Real-Time Updates:** The integration with PostgreSQL and Java Spring Boot allows for real-time updates and access to the latest information, which is vital during critical rescue operations.
- **Mobile Accessibility:** The mobile application developed with Flutter ensures that field users can access SAROP functionalities on the go, enhancing the flexibility and reach of the portal.

Evaluation

The evaluation of SAROP will be conducted through a series of tests and feedback sessions with actual search and rescue teams. The evaluation process includes:

- **Usability Testing:** Conducting usability tests to ensure that the portal is user-friendly and meets the needs of the team members. This includes testing the navigation, ease of operation entry, and map tracking features.
- **Performance Testing:** Assessing the performance of the portal under various conditions, including high traffic scenarios and offline mode. This helps in identifying any bottlenecks and ensuring that the system can handle the demands of real-world operations.
- **Security Testing:** Implementing security tests to ensure that the authentication and encryption mechanisms are robust and that the data is protected from unauthorized access.
- **Feedback Integration:** Collecting feedback from the users and incorporating their suggestions and improvements into the system. This iterative process ensures that SAROP evolves to meet the changing needs of search and rescue operations.
- **Impact Assessment:** Measuring the impact of SAROP on the efficiency and effectiveness of search and rescue operations. This includes tracking key metrics such as response time, success rates, and user satisfaction.

By conducting a thorough evaluation and continuously improving the system based on user feedback, we aim to ensure that SAROP becomes an indispensable tool for search and rescue teams, ultimately contributing to saving lives and reducing the impact of natural disasters.

For also future of SAROP, we aim to publish it on the Github and adding several features with new contributors. We plan of implementing features such as GPS tracking on maps, notification mechanism between the teams, users and operations, creating a role management system with permissions and creating different maps for each operation so that operations can be managed with maps in a more effective way.

5. Testing

TEST	PERFORMED MEMBER	PASSED/FAILED/FUTURE TEST
BACKEND- Authorization tests(Register/Login)	Mert Çıkla	PASSED
BACKEND- Team Location endpoints(Create/Read/Update/Delete)	Mert Çıkla	PASSED
BACKEND- Team Endpoints(Create/Read/Update/Delete)	Mert Çıkla	PASSED
BACKEND- Map Endpoints(Create/Read/Delete)	Mert Çıkla	PASSED
BACKEND- Geoserver Post/Delete Relation	Mert Çıkla	PASSED
BACKEND- BACKEND- Polygon Endpoints(Create/Read/Delete)	Mert Çıkla	PASSED
BACKEND- Note Endpoints(Create/Read/Delete)	Mert Çıkla	PASSED
BACKEND- Operation Endpoints(Create/Read/Update/Delete)	Mert Çıkla	PASSED
BACKEND- Category Endpoints(Create/Read/Update/Delete)	Mert Çıkla	PASSED
BACKEND- Exception Handling – Crashing case when there is an error	Mert Çıkla	PASSED
BACKEND -Exception Handling – Error messages	Mert Çıkla	Future Test
BACKEND- Loading Data From API	Mert Çıkla	PASSED
BACKEND- Performance test while loading data from API	Mert Çıkla	FAILED
BACKEND- Performance test for the relation with Geoserver API	Mert Çıkla	PASSED
MOBILE- User Authentication and Registration	Arda Gök	Passed
MOBILE- Registration screen works properly	Arda Gök	Passed
MOBILE Login with valid credentials	Arda Gök	Passed
MOBILE- Display of appropriate error messages	Arda Gök	Passed
MOBILE - Data security during registration/login	Arda Gök	Passed
MOBILE- Utilization of API Read function	Arda Gök	Passed
MOBILE- Proper handling of authentication	Arda Gök	Passed
MOBILE- Interaction between frontend and backend	Arda Gök	Passed
MOBILE- Accuracy of dropdown menu data fetching	Arda Gök	Passed
MOBILE- Data security during registration/login	Arda Gök	Passed
MOBILE- Display of appropriate error messages	Arda Gök	Passed

MOBILE- Proper response to incorrect requests	Arda Gök	Passed
MOBILE- API's response to errors and failures	Arda Gök	Passed
MOBILE- Usability of the user interface	Arda Gök	Passed
MOBILE- Smooth operation of all features	Arda Gök	FAILED
MOBILE- Easy navigation and execution of actions	Arda Gök	Passed
MOBILE- Accurate and quick map rendering	Arda Gök	FAILED(NOT QUICK)
MOBILE- Selection and visualization capabilities	Arda Gök	Passed
MOBILE- Functionality of each frontend component	Arda Gök	Passed
MOBILE- Response to user interactions	Arda Gök	Passed
MOBILE- Validation of error messages	Arda Gök	Passed
MOBILE- End-to-end user flow validation	Arda Gök	Passed
FRONTEND- User Authentication and Registration	Ceren Özdoğan	Passed
FRONTEND- Role-Based Access Control Validation	Ceren Özdoğan	Passed
FRONTEND- Interactive Map Display and Controls	Nursu Baltacı	Passed
FRONTEND- Map Marker Placement and Path Drawing	Nursu Baltacı	Future Test
FRONTEND- Operation Management Functions	Nursu Baltacı	Passed
FRONTEND- Real-Time Updates and Notifications	Ceren Özdoğan	Passed
FRONTEND- Cross-Browser Compatibility	Ceren Özdoğan	Passed - Tested on Chrome, Firefox, Safari
FRONTEND- Responsive Design on Various Devices	Ceren Özdoğan	Passed- Includes tests on desktops,tablets,smartphones
FRONTEND- User Interface Consistency	Ceren Özdoğan	Passed
FRONTEND- Accessibility Compliance Testing	Ceren Özdoğan	Future Test
FRONTEND- Security Measures for User Data	Nursu Baltacı	Passed
FRONTEND- Error Message Handling	Nursu Baltacı	Passed
FRONTEND- Session Management Testing	Nursu Baltacı	Passed
FRONTEND-Load Times and Performance Optimization	Nursu Baltacı	Failed - Needs optimization for faster load times
FRONTEND-Usability Testing	Ceren Özdoğan	Passed- Scheduled to gather user feedback on interface usability

6. Conclusion

Creating the Search and Rescue Operation Portal (SAROP) has been a difficult but worthwhile project. The sad events of the February 6 earthquake brought to light the urgent need for improved search and rescue operations administration and coordination, which served as the impetus for this project. In order to meet these objectives, SAROP offers a strong, user-friendly platform that improves the efficacy and efficiency of disaster response initiatives.

We have effectively combined several technologies throughout this project to produce an all-inclusive solution for search and rescue units. The backend, which was created with PostgreSQL and Java Spring Boot, guarantees a dependable and scalable architecture. The Flutter-developed mobile application increases the system's accessibility for field users, while the HTML, CSS, and JavaScript-implemented frontend offers an easy-to-use interface.

Our comprehensive testing and assessment procedures have shown that SAROP has the ability to greatly enhance the administration of search and rescue operations. During crucial rescue missions, the platform's features—like secure authentication, real-time updates, and advanced mapping capabilities—allow for better coordination and resource allocation. The system as a whole has shown to be dependable and efficient.

SAROP, in summary, is a major advancement in the use of technology for disaster relief. Our dedication to incorporating user feedback and making constant improvements will guarantee that SAROP continues to be an essential tool for search and rescue teams, ultimately helping to save lives and lessen the effects of calamities in the future.