הנדסת תוכנה
שנקר
מהנד0ים טאלנטים בת1כנה

# ScubAround

The scuba diving around you

## Software Design Document

Omer Portnoy 207251018, Dean Shalev 209707470

Shenkar College
Faculty of Engineering

Date: 19/07/2023

## Content

# 1. Introduction

In this System Design document, we present a comprehensive overview of the ScubAround application, detailing the architectural decisions, design considerations, and functional specifications that underpin the development of this online platform for scuba diving enthusiasts.

## 1.1. System Overview

Our application, ScubAround, is an online platform for scuba diving enthusiasts. It allows users to connect, plan dives, and share their diving experiences.

## 1.2. Problem Description

Scuba diving enthusiasts often face challenges in finding dive buddies, organizing dive trips, and accessing information about dive sites. ScubAround aims to address these challenges by providing a centralized platform for divers to connect, plan dives, and access dive-related information.

## 1.3. Goals

The goals of ScubAround are:

Facilitate connections between scuba divers to find dive buddies and form dive groups.

Provide a platform for divers to plan and organize dive trips, including selecting dive sites, scheduling dives, and managing logistics.

Enable divers to share their dive experiences, photos, and recommendations with the community.

Ensure a user-friendly and intuitive interface for easy navigation and interaction.

## 1.4. Scope

ScubAround will initially focus on serving scuba diving enthusiasts in specific geographic regions. The platform will include features for user registration, login, profile management, dive site listing, dive trip planning, and social interaction among users. However, it will not include e-commerce functionalities or certification verification.

## 1.5. Definitions and Acronyms

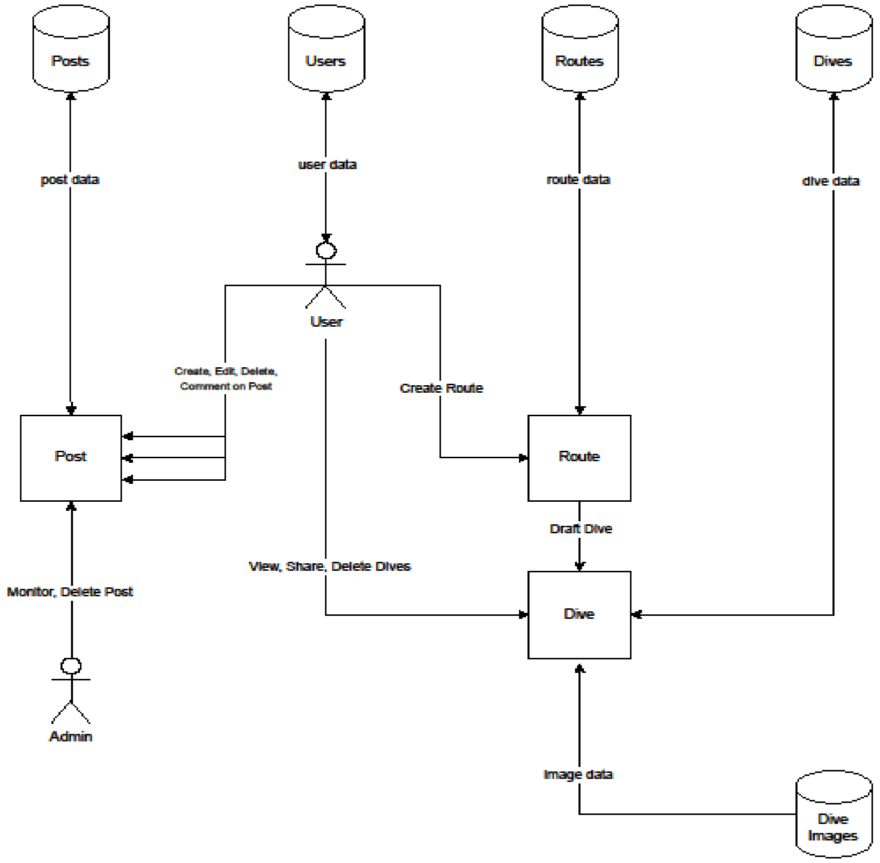
Dive Buddy: A fellow diver who accompanies another diver during a dive.

Dive Site: A specific location where scuba diving takes place, usually known for its unique features and underwater attractions.

Dive Trip: An organized excursion for scuba divers, including multiple dives at different dive sites.

## 2. System Architecture – System Context Diagram

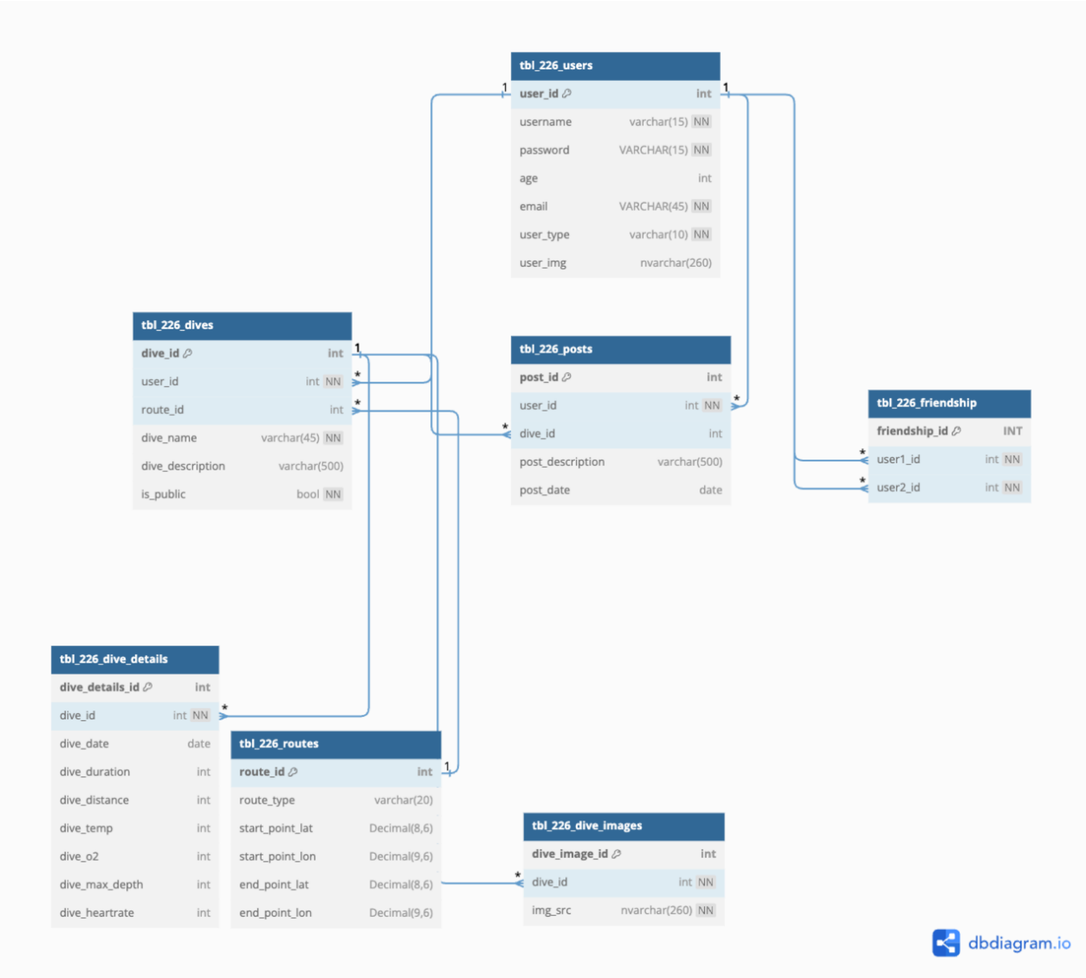
In this section, we will delve into the fundamental structure and components that constitute the ScubAround system, providing an overview of its high-level design and interactions.



# 3. System Design

The System Design section outlines the detailed specifications and functionality of the ScubAround application, presenting a comprehensive breakdown of its key features and user interactions.

## 3.1. **Data Design** - Database Description



| tbl_226_users | | |
|---|---|---|
| user_id | int | |
| username | varchar(15) | NN |
| password | VARCHAR(15) | NN |
| age | int | |
| email | VARCHAR(45) | NN |
| user_type | varchar(10) | NN |
| user_img | nvarchar(260) | |

| tbl_226_dives | | |
|---|---|---|
| dive_id | int | |
| user_id | int | NN |
| route_id | int | |
| dive_name | varchar(45) | NN |
| dive_description | varchar(500) | |
| is_public | bool | NN |

| tbl_226_posts | | |
|---|---|---|
| post_id | int | |
| user_id | int | NN |
| dive_id | int | |
| post_description | varchar(500) | |
| post_date | date | |

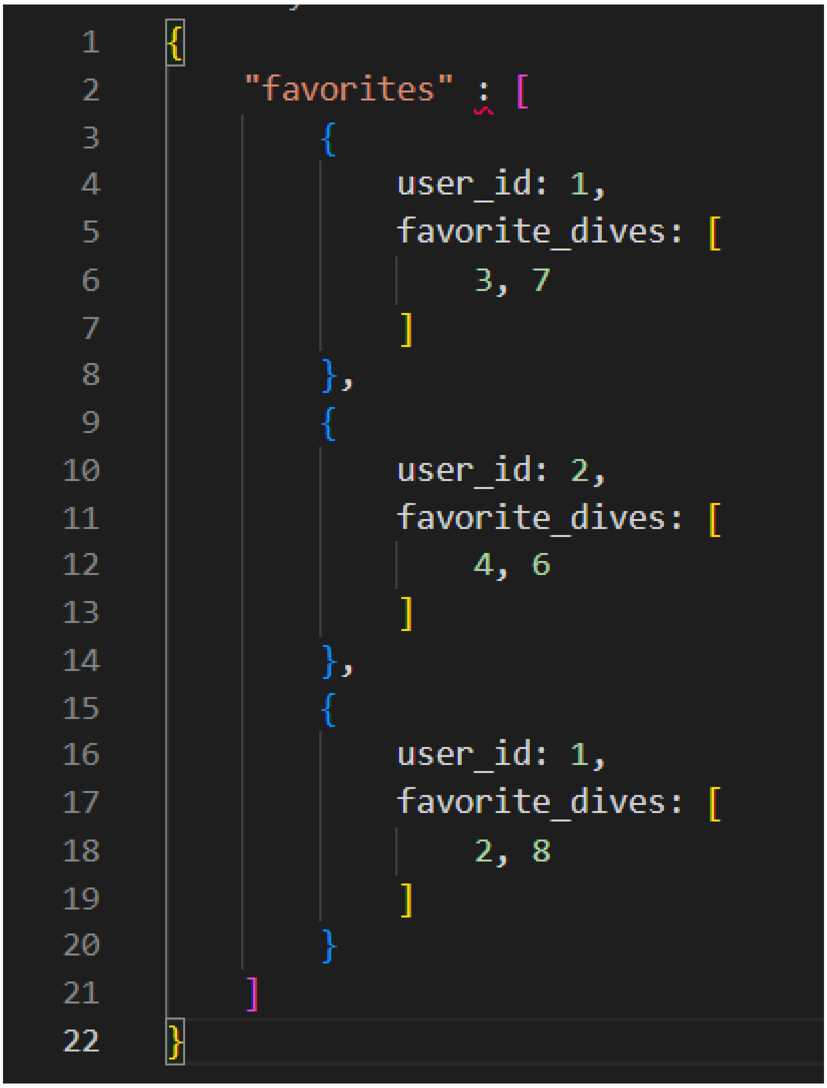
| tbl_226_friendship | | |
|---|---|---|
| friendship_id | INT | |
| user1_id | int | NN |
| user2_id | int | NN |

| tbl_226_dive_details | | |
|---|---|---|
| dive_details_id | int | |
| dive_id | int | NN |
| dive_date | date | |
| dive_duration | int | |
| dive_distance | int | |
| dive_temp | int | |
| dive_o2 | int | |
| dive_max_depth | int | |
| dive_heartrate | int | |

| tbl_226_routes | |
|---|---|
| route_id | int |
| route_type | varchar(20) |
| start_point_lat | Decimal(8,6) |
| start_point_lon | Decimal(9,6) |
| end_point_lat | Decimal(8,6) |
| end_point_lon | Decimal(9,6) |

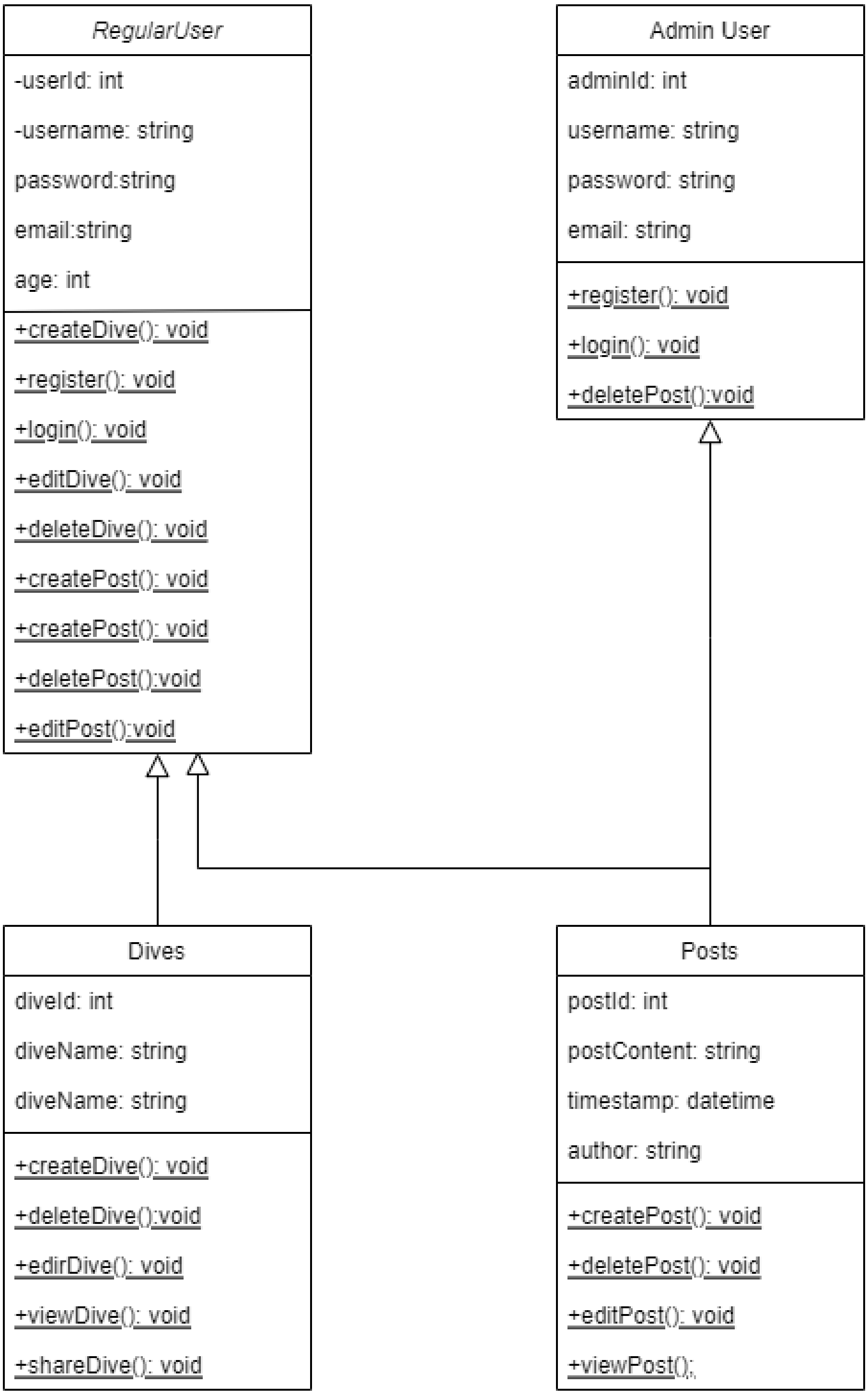
| tbl_226_dive_images | | |
|---|---|---|
| dive_image_id | int | |
| dive_id | int | NN |
| img_src | nvarchar(260) | NN |

JSON Structure:



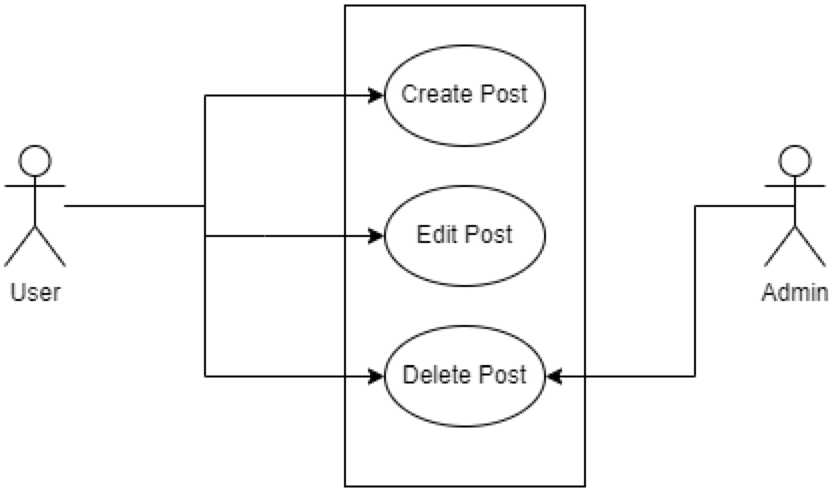
```
{
    "favorites" : [
        {
            user_id: 1,
            favorite_dives: [
                3, 7
            ]
        },
        {
            user_id: 2,
            favorite_dives: [
                4, 6
            ]
        },
        {
            user_id: 1,
            favorite_dives: [
                2, 8
            ]
        }
    ]
}
```

## 3.2. **Structural Design** - Class Diagram

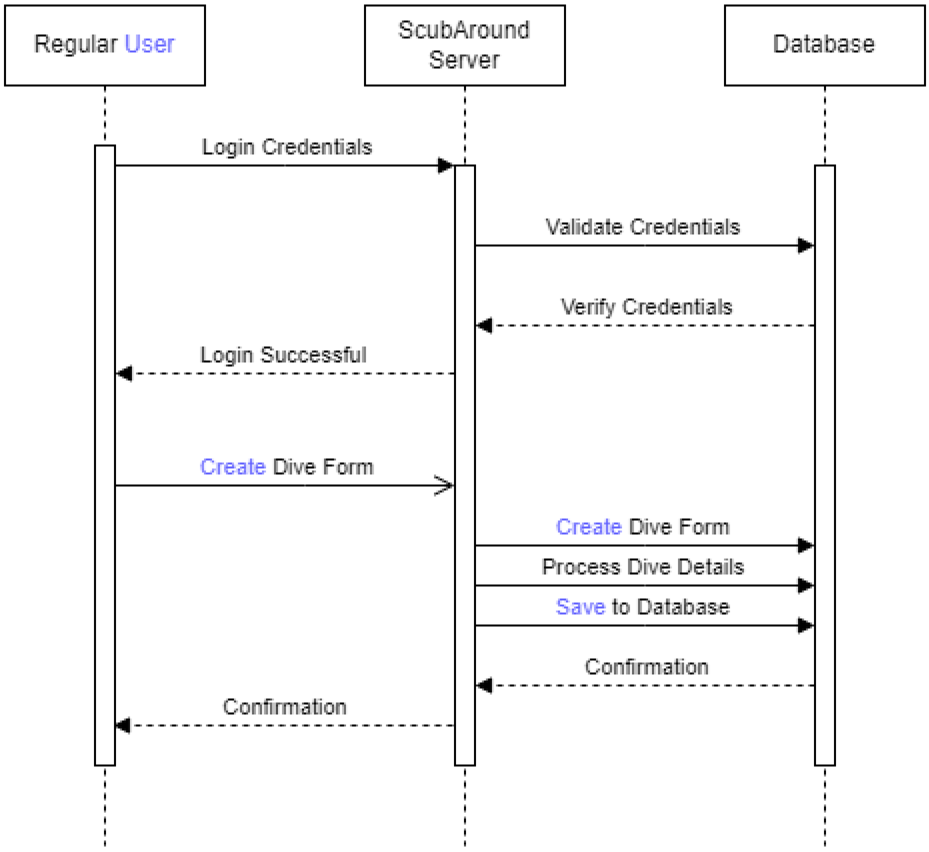


## 3.3. Interactions Design

### 3.3.1.Use Cases

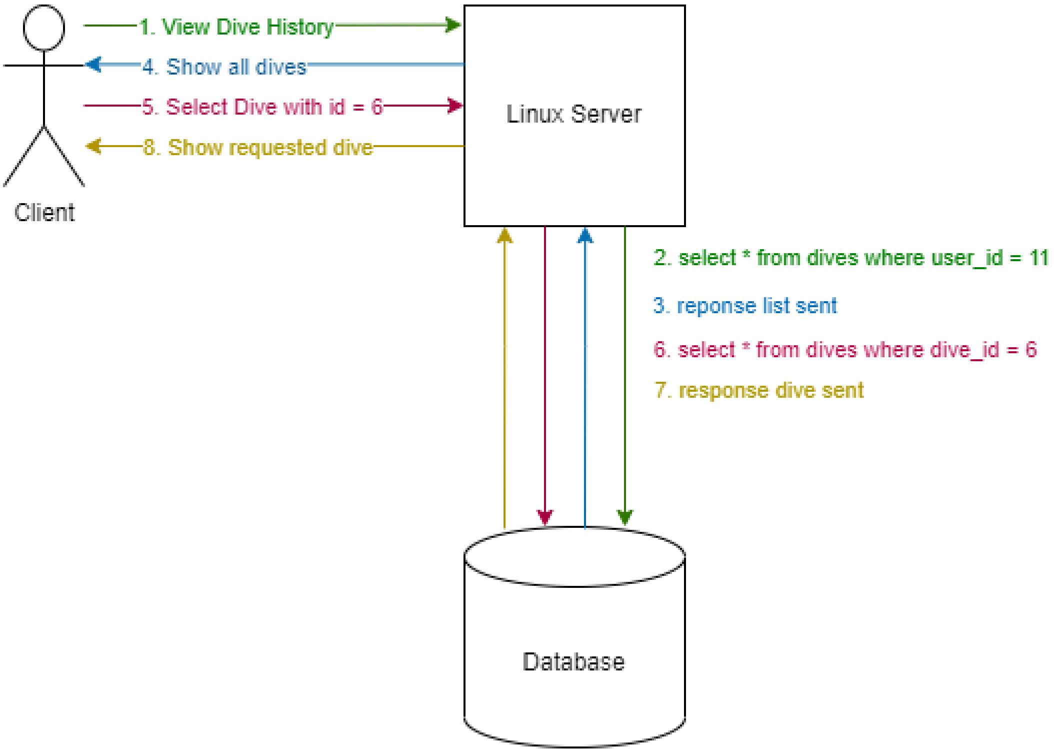


### 3.3.2.Sequence Diagram



### 3.4. Software Architecture Pattern

Client requests to view his dive history (all dives where user_id matches that of the clients), then wishes to view a certain dive (where dive_id matches the requested dive).



## 4. Verification and Validation

Functional Verification:

a. User Registration and Login:

Verify that users can successfully register with the system by providing necessary information like username, password, and email.
Validate that users can log in with their registered credentials and gain access to their accounts.

b. Create Dive Functionality:

Validate that Regular Users can create a new dive plan by providing relevant details such as dive name, location, date, and participants.
Verify that the system correctly processes the dive details and saves them to the database.

c. Create Post Functionality:

Verify that Regular Users can create a post by adding content and multimedia (e.g., images or videos).
Validate that the system correctly stores the post data in the database.

d. Delete Post Functionality (Admin User privilege):

Verify that Admin Users can delete posts created by Regular Users, ensuring that only Admins have this privilege.

e. View Dive Functionality:

Validate that users can view the details of a specific dive, including its name, location, date, and participants.

f. View Post Functionality:

Verify that users can view posts shared by others, including the post content and author details.

g. Edit Dive Functionality (Regular User privilege):

Validate that Regular Users can edit the details of a dive plan they created/performed, such as updating the dive name, location, or date.

h. Security Verification:

Verify that the system has proper authentication and authorization mechanisms in place to protect user data and prevent unauthorized access.

i. Compatibility Verification:

Validate that the ScubAround system works on different web browsers and devices to ensure cross-platform compatibility.

j. Usability Verification:

Conduct usability testing to ensure that the user interface is intuitive, user-friendly, and easy to navigate.