CoastalGuard

A PROJECT REPORT

Submitted by

MUTHU KARTHIK A	953021104031
ESAKKI PREM KUMAR M	953021104012
KARUNAKARAN S	953021104022

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ST. MOTHER THERESA ENGINEERING COLLEGE, VAGAIKULAM

ANNA UNIVERSITY : CHENNAI 600 025

JUNE 2025

BONAFIDE CERTIFICATE

Certified that this project report **“CoastalGuard”** is the bonafide work of **“A.MUTHUKARTHIK, M.ESAKKI PREM KUMAR , S.KARUNAKARAN”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Ms. M. DANI PACKIASEELI AP/CSE.,	M.BEEMAJAN SHAHEEN, M.Tech.,
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor
Computer Science and Engineering	Computer Science and Engineering
St. Mother Theresa Engineering College	St. Mother Theresa Engineering College
Vagaikulam, Tuticorin-628102	Vagaikulam, Tuticorin-628102

Submitted for the viva voice examination held on ………………….

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

With pleasure we express our profound gratitude to our beloved chairman **Dr.S.Cletus Babu** and also to our voice chairperson **Dr.S.Amali Cletus** for giving motivation and providing all necessary facility for the successful completion of this project.

It is our privilege to thank our beloved General Manager **Dr.A.George Klington** for his moral support and deeds in bringing out this project successfully.

We would like to express our special thanks to our Principal, **Dr.K.M.MuthuKrishnan** for his time and efforts he provided throughout the year.

We would like to thank, **M.Beemajan Shaheen**, M.Tech.,Head of the Department, Department of Computer Science And Engineering and our Guide without whom we would not have pursued this course. Her helpful hands in each step of our project and for her guidance and constant support in the successful completion and improvement of the project.

We express our thanks to all of our department **staff members, friends and family members** for their advice and encouragement to do the project work with full interest and enthusiasm.

CoastalGuard: Enhancing Maritime Border Security Through Real-Time Surveillance and Predictive Alerts

Abstract

Maritime security is of paramount importance for coastal nations, given the numerous challenges posed by illegal border crossings, smuggling activities, and environmental hazards. Traditional surveillance methods have limitations in effectively monitoring vast maritime borders in real time and predicting potential security breaches. Additionally, fishermen operating in these areas are often exposed to risks such as adverse weather conditions and maritime hazards due to limited access to timely information. Hence, there is a critical need for innovative technological solutions to enhance border surveillance, communication, and coordination among maritime authorities and stakeholders. In response to these challenges, the aim of the project is to develop a solution to address the shortcomings of traditional border surveillance systems This project introduces an integrated border alert system that combines weather monitoring, hazard alerts, and seamless integration with maritime authorities. Leveraging Long Short-Term Memory (LSTM) neural networks, the FBAS aims to bolster surveillance and response capabilities. The BorderNet Model, is going to develop with LSTM neural networks, enables predictive border classification, while the Alert System issues timely notifications to stakeholders. The Weather Data Provider API ensures access to up-to-date meteorological information, enhancing decision-making. Additionally, features such as alert systems for adverse weather conditions and maritime hazards ensure the safety of fishermen and other maritime activities. Through its innovative features and capabilities, the proposed system aims to bridge the gaps in traditional surveillance systems and provide comprehensive protection for maritime borders.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. OVERVIEW

Fishing is the activity of trying to catch fish. Fish are often caught as wildlife from the natural environment (fresh water or marine), but may also be caught from stocked bodies of water such as ponds, canals, park wetlands and reservoirs. Fishing techniques include hand-gathering, spearing, netting, angling, shooting and trapping, as well as more destructive and often illegal techniques such as electrocution, blasting and poisoning. The term fishing broadly includes catching aquatic animals other than fish, such as crustaceans (shrimp/lobsters/crabs), shellfish, cephalopods (octopus/squid) and echinoderms (starfish/sea urchins). The term is not normally applied to harvesting fish raised in controlled cultivations (fish farming).

Fishing has been an important part of human culture since hunter-gatherer times. It is one of the few food production activities that has persisted from prehistory into the modern age, surviving both the Neolithic Revolution and successive Industrial Revolutions. In addition to being caught to be eaten for food, fish are caught as recreational pastimes. India has a coastline of 7,517 km, of which the mainland accounts for 5,422 km.

1.2. PROBLEM STATEMENT

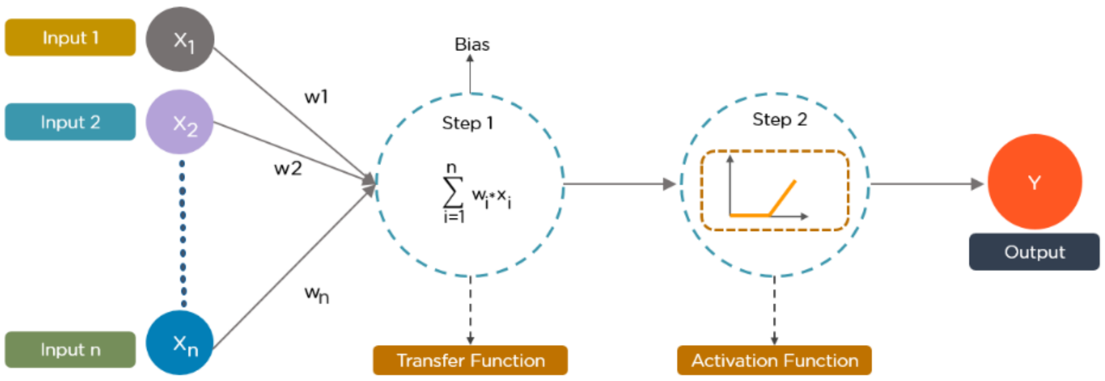
The Fisherman Border Surveillance System is a pivotal element of maritime security strategies, dedicated to monitoring and fortifying maritime borders, particularly in regions susceptible to illicit activities and security risks. It encompasses a multifaceted array of technologies, protocols, and methodologies aimed at detecting, deterring, and responding to unauthorized border crossings by fishing vessels. It was relied on a combination of manual surveillance techniques and technological aids. Manual surveillance entails the deployment of coast guard personnel, stationed at lookout points or aboard patrol vessels, to visually scan the maritime domain for suspicious activities. These personnel rely on their training and experience to discern potential threats and breaches of maritime borders. Technological aids play a pivotal role in bolstering manual surveillance efforts and extending the reach of border monitoring capabilities.

1.3. DEEP LEARNING

Deep learning is the most hyped branch of machine learning that uses complex algorithms of deep neural networks that are inspired by the way the human brain works. DL models can draw accurate results from large volumes of input data without being told which data characteristics to look at. Imagine you need to determine which fishing rods generate positive online reviews on your website and which cause the negative ones. In this case, deep neural nets can extract meaningful characteristics from reviews and perform sentiment analysis.

1.3.1. Importance of Deep Learning

Deep learning algorithms play a crucial role in determining the features and can handle the large number of processes for the data that might be structured or unstructured. Although, deep learning algorithms can overkill some tasks that might involve complex problems because they need access to huge amounts of data so that they can function effectively. For example, there's a popular deep learning tool that recognizes images namely **Imagenet** that has access to **14 million** images in its dataset-driven algorithms.



A neural network is structured like the human brain and consists of artificial neurons, also known as nodes. These nodes are stacked next to each other in three layers:

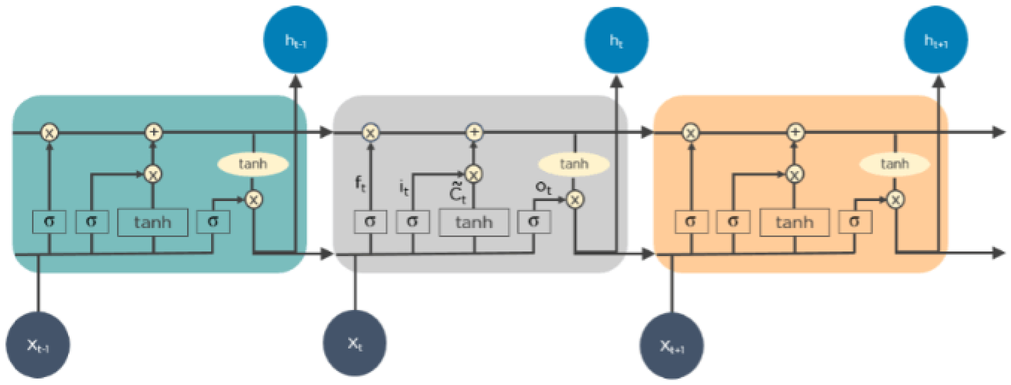
- The input layer
- The hidden layer(s)
- The output layer

Data provides each node with information in the form of inputs. The node multiplies the inputs with random weights, calculates them, and adds a bias. Finally, nonlinear functions, also known as activation functions, are applied to determine which neuron to fire.

1.3.2. Long Short Term Memory Networks (LSTMs)

LSTMs can be defined as Recurrent Neural Networks (RNN) that are programmed to learn and adapt for dependencies for the long term. It can memorize and recall past data for a greater period and by default, it is its sole behaviour.. This analogy comes from their **chain-like** structure consisting of **four** interacting layers that communicate with each other differently. Besides applications of time series prediction, they can be used to construct **speech recognizers, development in pharmaceuticals,** and composition of **music loops** as well.

LSTM work in a sequence of events. First, they don't tend to remember irrelevant details attained in the previous state.



1.4. AIM AND OBJECTIVE

Aim

The aim of the project is to enhance maritime security along the Tamil Nadu to Sri Lanka border through the development and implementation of an integrated system that leverages data-driven approaches and predictive analytics to proactively monitor, detect, and respond to potential border breaches by fishing vessels.

Objectives

- To develop a user friendly web application for maritime security operations.
- To integrate deep learning for predictive border classification.
- To collect and preprocess diverse datasets for model training.
- To implement a real-time alerting system for immediate notifications.

1.5. SCOPE OF THE PROJECT

The detailed scope of the project encompasses several key aspects:

- **Coast Guard Stations Web App Development**: The primary focus is on developing a web application using Python and Flask for the backend, MySQL for database management, and Bootstrap for the frontend. The app serves as a centralized platform for coast guard personnel to manage border security operations efficiently.
- **Integration of Machine Learning Models**: The project involves integrating machine learning capabilities using TensorFlow and Keras for predictive analytics. This includes developing models like BorderNet for classifying borders as safe or breached, enhancing the app's ability to predict and respond to potential threats.
- **Deployment of BorderAlert Devices**: Integrating BorderAlert GPS devices on fishing vessels enables continuous location tracking. This feature ensures that real-time vessel positions and trajectories are available to coast guard authorities, enhancing situational awareness and response capabilities.
- **User Authentication and Access Control**: Implementing robust authentication mechanisms ensures secure access to the web app. User management features allow coast guard personnel to control access levels, ensuring that only authorized users can perform specific actions within the system.
- **Real-time Weather Updates**: Integrating a weather data provider API enables the app to provide real-time meteorological data. This feature equips coast guard personnel with information on adverse weather conditions, enabling informed decision-making and proactive risk mitigation.
- **Hazard Alert Systems**: Implementing hazard alert systems notifies fishermen about potential dangers in their vicinity. Alerts are generated based on data from maritime surveillance systems or AIS, enhancing safety at sea by providing timely warnings about navigational hazards or other risks.

CHAPTER 2
LITERATURE SURVEY

2.1. IoT Assisted Fisherman Aid to Detect Borders and Alert System by using Intelligent GPS Technology

Author: A P. Ramesh; R. Nithya Paranthaman;

Year2024

Doi: 10.1109/ICSES60034.2023.10465484

Problem

The safety of fishermen faces significant challenges due to the lack of effective monitoring and communication systems.Traditional methods fall short in providing timely interventions, necessitating the integration of advanced technologies such as Global Positioning System (GPS) and Wireless Sensor Networks (WSN) to enhance safety measures.

Objective

This project aims to improve the safety of fishermen by utilizing GPS and WSN technologies to enable real-time tracking, monitoring, and communication. Specifically, the objectives include developing a robust GPS-based tracking system for precise location monitoring, implementing a WSN for seamless communication between fishermen and a centralized system, creating an alert mechanism to warn fishermen approaching restricted areas, and designing an automated response system to mitigate risks and ensure prompt actions during emergencies.

Methodology

The methodology involves a comprehensive approach comprising research, development, testing, and deployment phases. Initially, extensive research is conducted to understand the requirements and challenges faced by fishermen and to identify suitable technologies. Customized software and hardware solutions are developed based on the research findings to meet the specific needs of the fishing industry.

Finding

The implementation and testing of the integrated system demonstrated significant improvements in the safety and security of fishermen. Real-time tracking and communication capabilities provided by GPS and WSN technologies enabled proactive monitoring and rapid response to

2.2. Arduino based Maritime Border Alert System for Fisherman

Author: Lijo Jacob Varghese; A. Sakthivel

Year2022

Doi: 10.1109/ICICT54344.2022.9850676

Problem

Fishermen face precarious livelihoods, where accidental border crossings are treated as severe offenses, exacerbating their vulnerability. The situation escalates tragically when impoverished fishermen face violence, including being shot at and having their boats confiscated.

Objective

The primary aim is to develop a Maritime Border Alert System (MBAS) that aids fishermen by providing timely information about national borders, thereby safeguarding them and their vessels. This system intends to mitigate the risks associated with accidental border crossings and the subsequent punitive actions

Methodology

The proposed framework entails the development of a low-cost Maritime Boundary Crossing Warning System tailored specifically for fishermen. Initially, thorough research will be conducted to understand the challenges faced by fishermen and to identify suitable technologies.

Finding

The implementation of the Maritime Border Alert System is anticipated to significantly enhance the safety and security of fishermen operating in border regions. By providing real-time information about maritime borders, the system will empower fishermen to navigate safely and avoid inadvertent violations.

2.3. Alert system for fishermen crossing border using Android

Author: R Dinesh Kumar; M Shubin Aldo;

Year2022

Doi: 10.1109/ICEEOT.2016.7755630

Problem

Navigating border regions poses challenges for individuals, particularly those at sea, who require a reliable means to determine safe pathways. Additionally, conflicts with opposing forces further compound these challenges, necessitating proactive incident management solutions. Tamil fishermen, in particular, face heightened risks due to their employment in border areas.

Objective

The primary objective is to develop an application that facilitates safe navigation and incident management for individuals operating in border regions, with a focus on Tamil fishermen. The application will utilize device tracking technology to provide real-time location updates and notifications to both users and border security forces.

Methodology

The methodology involves the development of an intuitive application that operates on device tracking technology. Thorough research will be conducted to understand the unique requirements and challenges faced by individuals, particularly Tamil fishermen, operating in border regions. The application will be designed to be user-friendly, ensuring accessibility for individuals of all literacy levels

Finding

Upon implementation, the application is expected to significantly enhance the safety and security of individuals navigating border regions, particularly Tamil fishermen. Real-time location updates and incident notifications will empower users to make informed decisions and avoid potential conflicts.

2.4. Fisherman Border Detector Using IOT Andlora

Author: Adhithyan K; S M Jainth;

Year2023

Doi: 10.1109/ICCEBS58601.2023.10448653

Problem

The lives of fishermen in peninsula, island, and coastal countries are endangered due to their unawareness of maritime boundary limits, resulting in accidental border crossings. This ignorance exposes them to various risks, including abduction and confiscation of their boats by neighboring militants. Tragically, many fishermen have lost their lives due to conflicts arising from these border violations.

Objective

The primary objective of this project is to develop a system that effectively alerts fishermen about maritime border limits, thereby preventing accidental border crossings and minimizing risks to their lives and livelihoods. The system aims to utilize GPS technology to provide real-time information to fishermen regarding their proximity to maritime borders.

Methodology

The methodology involves the implementation of a GPS-based alert system designed to notify fishermen about their distance from maritime borders. Thorough research will be conducted to understand the specific needs and challenges faced by fishermen operating in border regions.

Finding

Upon implementation, the system is expected to significantly enhance the safety and security of fishermen operating in border regions. By providing real-time alerts about maritime border limits, the system will enable fishermen to navigate safely and avoid accidental border crossings.

2.5. Border Security System for Fishermen using Radio Frequency Technique with Live Streaming

Author: V. Sujitha; V. Anandhan

Year2023

Doi: 10.1109/ICAAIC56838.2023.10140412

Problem

The primary challenge contributing to border-cross brutality is the ambiguity surrounding water borders between neighboring nations. This ambiguity exposes fishermen to grave risks, as they frequently encounter gunfire from opposing fleets.

Objective

This research aims to address the safety concerns of fishermen operating in disputed maritime borders by developing an embedded system-based device. The device leverages Global Positioning System (GPS) and radio frequency technology to alert border authorities to fishermen's whereabouts in real-time.

Methodology

The methodology involves the design and development of an embedded system-based device tailored to the specific needs of fishermen operating in disputed maritime borders. Thorough research will inform the selection and integration of GPS and radio frequency technologies to ensure accurate location tracking and communication capabilities.

Finding

Upon implementation, the embedded system-based device is expected to significantly enhance the safety and security of fishermen navigating disputed maritime borders. Real-time location tracking and communication capabilities provided by GPS and radio frequency technologies will enable swift response from border authorities in the event of emergencies or altercations.

2.7. Border alert system and emergency contact for Fisherman using RSSI

Author: E. Krishnamoorthy; S. Manikandan

Year2017

Doi: 10.1109/ICICES.2017.8070766

Problem

Tensions along ocean borders have led to significant conflict between two nations, resulting in devastating consequences. Fishermen from either country are frequently apprehended by naval forces for inadvertently crossing the border.

Objective

The primary goal of this system is to track the location of fishing vessels using Received Signal Strength Indication (RSSI) technology, providing reliable navigation and timing services to users worldwide.

Methodology

The system employs RSSI technology to track the location of fishing vessels in real-time, ensuring accurate navigation and timing services. Upon approaching or crossing a maritime border, the system triggers an advisory message to alert fishermen.

Finding

Implementation of this system is expected to significantly reduce incidents of fishermen inadvertently crossing maritime borders and facing apprehension by naval forces. The use of RSSI technology ensures accurate and reliable tracking of fishing vessels, enabling timely warnings to be issued to fishermen.

CHAPTER 3
SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

The existing system of fisherman border surveillance relies on conventional methods and technologies, which differ significantly from modern, technology-driven approaches. Here's an outline of the traditional system:

- **Visual Surveillance**

Historically, fisherman border surveillance predominantly involved visual monitoring by coast guard personnel stationed at lookout points along the coastline .

- **Manual Patrols**

Coast guard patrols using boats or vessels are another traditional method of fisherman border surveillance. These patrols involve physically patrolling the maritime borders to deter and intercept any unauthorized vessels attempting to cross the borders.

- **Radio Communication**

Communication between coast guard stations and fishing vessels traditionally occurs via marine VHF radio. Fishermen often report their positions and activities to coast guard authorities using radio channels designated for maritime communication.

- **Radar and AIS Integration**

Radar systems and Automatic Identification System (AIS) technology are integrated into the existing surveillance infrastructure to track vessel movements in real-time. Radar systems detect vessels within the surveillance area, while AIS transponders on fishing vessels transmit vessel identification and position data, enabling authorities to monitor maritime traffic effectively.

- **GPS Tracking**

Fishing vessels are equipped with GPS tracking devices that provide precise location data, allowing coast guard authorities to monitor vessel movements and ensure compliance with designated fishing zones and maritime borders. GPS tracking enhances situational awareness and enables timely response to border breaches or emergencies.

Existing Algorithms

The existing algorithms used in fisherman border alert systems encompass a variety of methods aimed at detecting and responding to potential security threats and breaches along maritime borders. Here are some of the key algorithms commonly employed in such systems:

- **Pattern Recognition Algorithms**

Pattern recognition algorithms analyze historical data and surveillance information to identify patterns indicative of suspicious or unauthorized activities, such as abnormal vessel movements or deviations from typical fishing routes. Machine learning techniques, including clustering, classification, and anomaly detection algorithms, are often used to recognize patterns and anomalies in vessel behavior.

- **Route Prediction Algorithms**

Route prediction algorithms utilize historical vessel trajectory data and predictive analytics to forecast the future movements and routes of fishing vessels. By analyzing past behavior and environmental factors, these algorithms can predict the probable paths that vessels are likely to follow, enabling authorities to anticipate potential border crossings or deviations from expected routes.

- **Data Fusion and Fusion Algorithms**

Data fusion and fusion algorithms integrate information from diverse sources, including radar, AIS, GPS, satellite imagery, and sensor networks, to generate a comprehensive picture of maritime activities and border conditions. These algorithms employ fusion techniques such as Bayesian inference, Dempster-Shafer theory, and Kalman filtering to combine heterogeneous data streams and enhance situational awareness for decision-making.

- **Random Forest**

Random Forest is an ensemble learning algorithm that combines multiple decision trees to classify fishing vessel behavior and detect border breaches. Random Forest can handle large and complex datasets and is effective in capturing nonlinear relationships and interactions between features in maritime surveillance data.

- **K-Nearest Neighbors (KNN)**

KNN is a non-parametric algorithm used for classification and anomaly detection in fisherman border alert systems. KNN classifies fishing vessel behavior based on the similarity of their feature vectors to those of neighboring instances in the dataset..

3.2. PROPOSED SYSTEM

- **Coast Guard Stations Web App**

The centerpiece of the system, this web application provides a user-friendly interface for coast guard personnel to manage border security operations. It includes features for real-time border surveillance, communication with authorities, and coordination with fishermen.

- **BorderNet Model**

The BorderNet Model is a deep learning system that employs LSTM neural networks to classify maritime borders as either safe or breached. It analyzes various data, including vessel trajectories and environmental conditions, to predict border status in real-time.

3.2.1. ADVANTAGES

- Real-time border monitoring enhances security.
- Predictive analytics for proactive border breach prevention.
- User-friendly interface for easy access by coast guard and fishermen.
- Timely alerts for breaches, weather, and hazards prompt action.
- Fishermen safety through hazard alerts.

3.3. FEASIBILITY STUDY

The feasibility analysis of the project assesses its technical, economic, and operational viability. Here's an overview:

3.3.1. Technical Feasibility

- The project requires expertise in Python, Flask, MySQL, TensorFlow, Keras, and other relevant technologies, which are readily available and well-documented.
- The availability of machine learning libraries and frameworks supports the implementation of predictive analytics for border surveillance.
- The use of Wampserver for local server hosting provides a convenient development environment, although cloud hosting options should be considered for scalability.

3.3.2. Financial Feasibility

- The project incurs initial costs for hardware, software licenses, and personnel training, which are manageable given the availability of open-source tools and frameworks.
- Operational costs, including server maintenance, data storage, and potential subscription fees for cloud services, are relatively low compared to the potential benefits of enhanced maritime security.

CHAPTER 4
SYSTEM CONFIGURATION

4.1. HARDWARE SPECIFICATIONS

- **Processor:** Multi-core CPU for handling concurrent requests and computations.
- **RAM:** 16GB RAM to accommodate data processing and model training tasks.
- **Storage:** 256 SSD (Solid State Drive) storage with ample space for storing datasets, models, and system files.

4.2. SOFTWARE SPECIFICATIONS

- **Operating System:** Windows 10 or 11 (for Windows-specific development)
- **Programming Language:** Python (version 3.6 or higher)
- **Neural Network Framework:** TensorFlow
- **Data Processing Libraries**: Pandas, NumPy, Scikit Learn
- **Visualization Libraries**: Matplotlib, Seaborn
- **Web Framework:** Flask for implementing a web-based user interface,
- **Database Integration:** MySQL for storing and retrieving relevant data.
- **Integrated Development Environment (IDE):** IDLE
- **Web Technologies:** HTML, CSS, and JavaScript

CHAPTER 5

SOFTWARE DESCRIPTION

5.1. PYTHON 3.8

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

Tensor Flow

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and gives developers the ability to easily build and deploy ML-powered applications.

TensorFlow provides a collection of workflows with intuitive, high-level APIs for both beginners and experts to create machine learning models in numerous languages.

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Keras is an API designed for human beings, not machines.

Keras 3 is a multi-backend deep learning framework, with support for JAX, TensorFlow, and PyTorch.

Pandas

pandas are a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. pandas are a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive.

Pandas is mainly used for data analysis and associated manipulation of tabular data in Data frames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Scikit Learn

scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

Scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

5.2. MYSQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are:

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database.

5.3. WAMPSERVER

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world.

WAMPServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features numerous functionalities and makes it the preferred choice of developers from around the world.

WAMP Server Features

- Apache Webserver
- MySQL DB Server
- MariaDB Server
- PHP Scriting language installed
- WAMP Server Tools
- PhpMyAdmin to manage DBs
- Manage Apache and MySQL services
- Switch to online / offline mode (accessible to all or limited to localhost)
- Install and change version of Apache, MySQL and PHP
- Manage the configuration parameters of your servers
- Access your logs
- Access configuration fi

5.4. BOOTSTRAP 4

Bootstrap is a powerful front-end framework for faster and easier web development. Bootstrap is a free and open-source web development framework. It’s designed to ease the web development process of responsive, mobile-first websites by providing a collection of syntax for template designs. In other words, Bootstrap helps web developers build websites faster as they don’t need to worry about basic commands and functions

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework
Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

5.5. FLASK

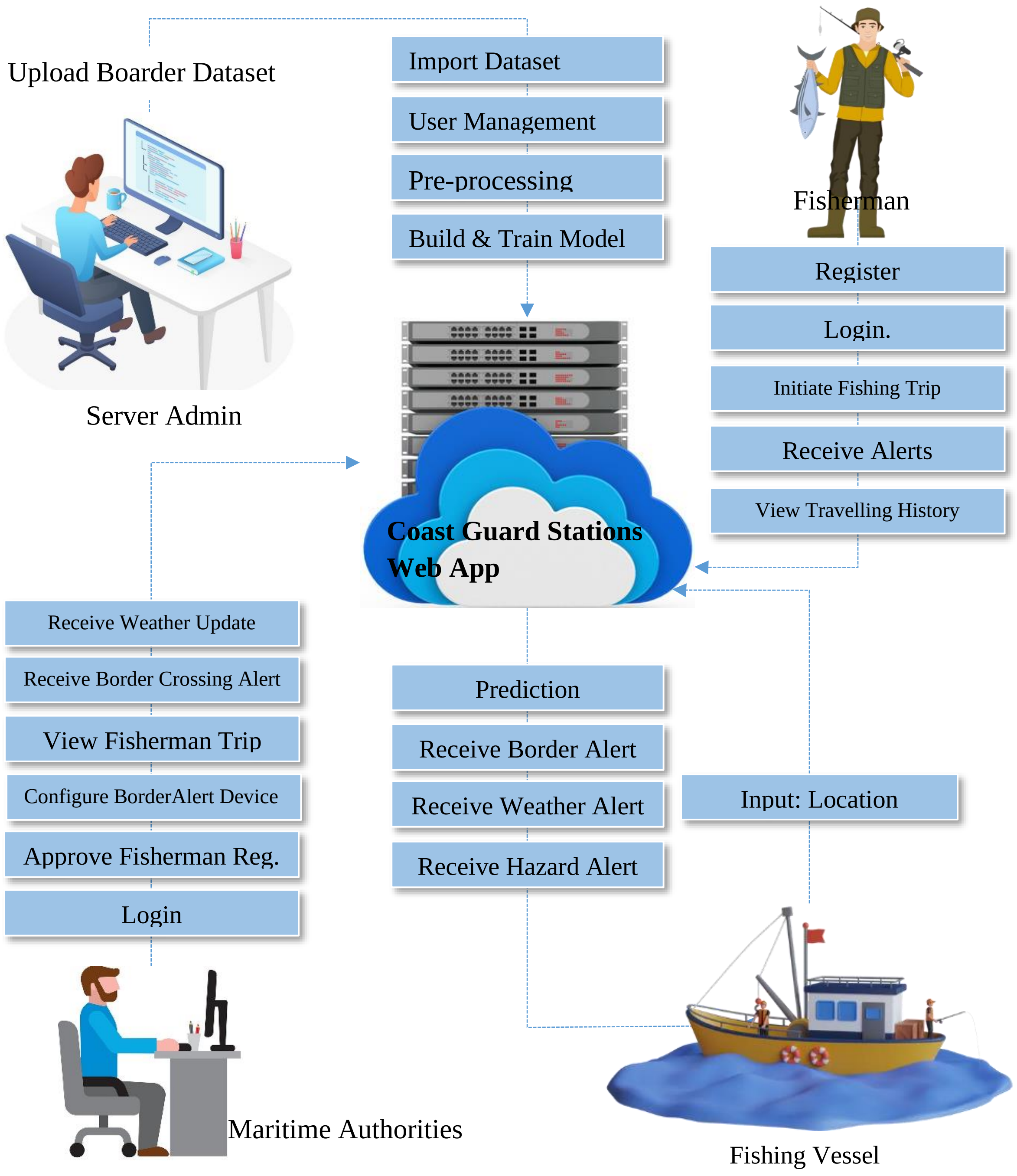
Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers.

CHAPTER 6

SYSTEM DESIGN

6.1. SYSTEM ARCHITECTURE



6.1. INPUT DESIGN

1. **User Authentication**
    - Input fields for username and password to authenticate users accessing the system.
2. **Registration Form**
    - Input fields for fishermen to provide necessary identification details such as Aadhar number and Fisherman Card details.
3. **Border Observation Data Entry**
    - Input fields to capture border observation data, including timestamp, latitude, longitude, vessel speed, direction, weather conditions, temperature, sea state, and marine traffic.
4. **Alert Configuration**
    - Input fields or checkboxes for users to customize alert preferences, including notification channels (e.g., mobile app, SMS, email) and threshold settings for triggering alerts.
5. **Data Import**
    - Support for various file formats such as CSV, Excel, or JSON, with error handling for incomplete or corrupted files.
6. **Profile Management**
    - Input fields to update user profiles, including contact information, vessel details, and emergency contacts.
7. **Weather Data Input**
    - Integration with weather APIs to automatically fetch real-time meteorological data.
8. **BorderNet Model Training Data Input**
    - Input interface for uploading training datasets containing historical border observation data.

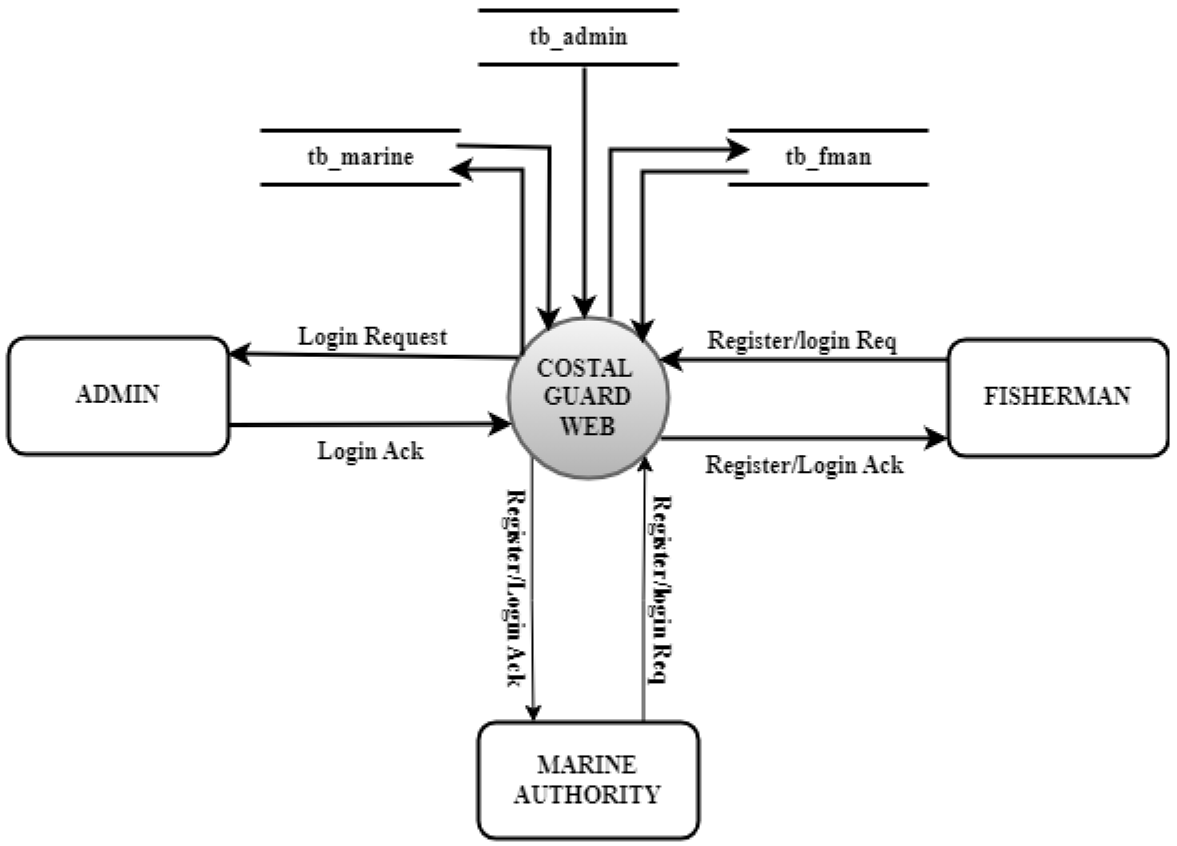
6.2. OUTPUT DESIGN

1. **Border Surveillance Dashboard**
    - Visual representation of real-time border surveillance data, including vessel
2. **Alert Notifications**
    - Instant alerts sent to users via preferred notification channels (e.g., mobile app, SMS, email) in response to detected border breaches, adverse weather conditions, or maritime hazards.

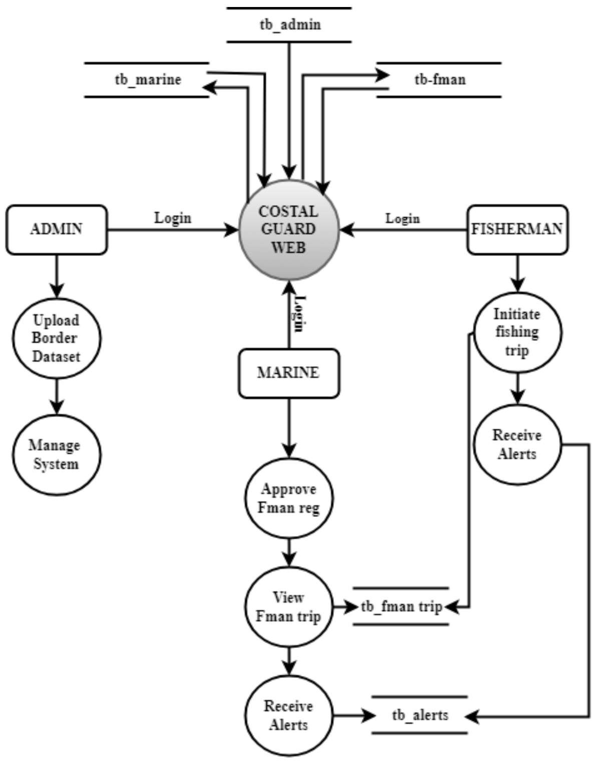
3. **Predictive Analytics Reports**
    - Reports generated by the BorderNet model showcasing predictions and forecasts of potential border breaches based on historical data and machine learning algorithms.
4. **User Profile Updates**
    - Confirmation messages or status indicators displayed upon successful updates to user profiles, including contact information, vessel details, and alert preferences.
5. **Weather Forecasts**
    - Display of real-time and forecasted meteorological data, including temperature, wind speed, precipitation, and sea conditions.
6. **BorderNet Model Training Results**
    - Feedback on the training process of the BorderNet model, including training accuracy, loss metrics, and convergence status.

6.4. DATA FLOW DIAGRAM

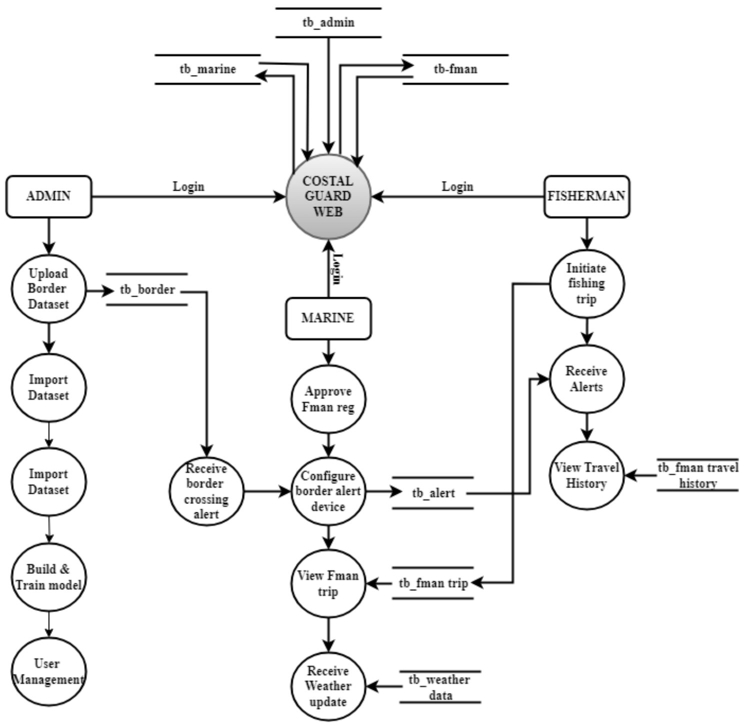
LEVEL 0



LEVEL 1

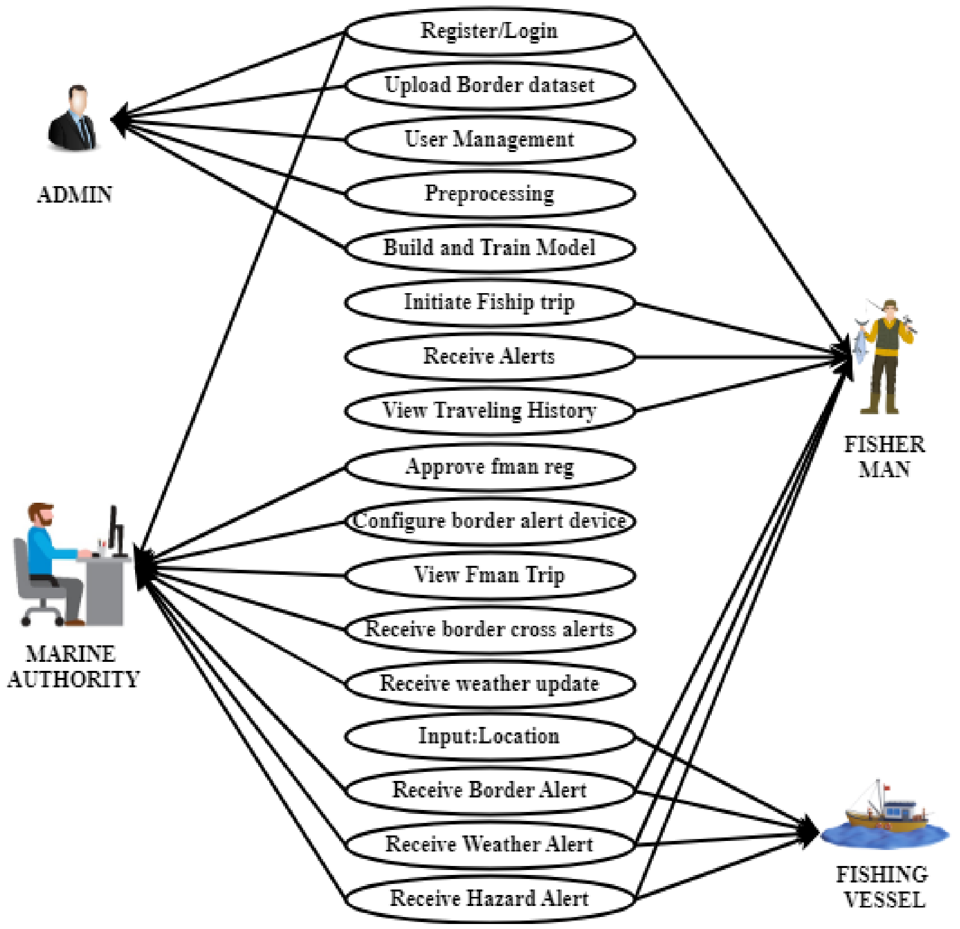

tb_admin
tb_marine
tb-fman
ADMIN
Login
COSTAL GUARD WEB
Login
FISHERMAN
Login
Upload Border Dataset
Manage System
MARINE
Approve Fman reg
View Fman trip
tb_fman trip
Receive Alerts
tb_alerts
Initiate fishing trip
Receive Alerts

LEVEL 2

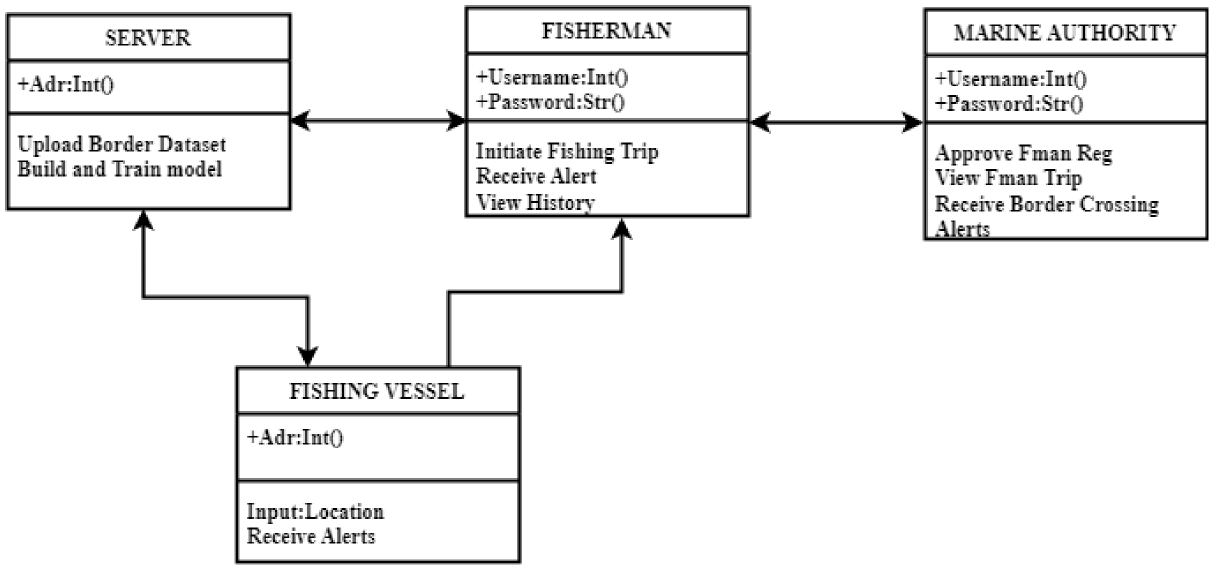

tb_admin
tb_marine
tb-fman
ADMIN
Login
COSTAL GUARD WEB
Login
FISHERMAN
Upload Border Dataset
tb_border
Import Dataset
Import Dataset
Build & Train model
User Management
Receive border crossing alert
Login
MARINE
Approve Fman reg
Configure border alert device
tb_alert
View Fman trip
tb_fman trip
Receive Weather update
tb_weather data
Initiate fishing trip
Receive Alerts
View Travel History
tb_fman travel history

6.5. UML DIAGRAM

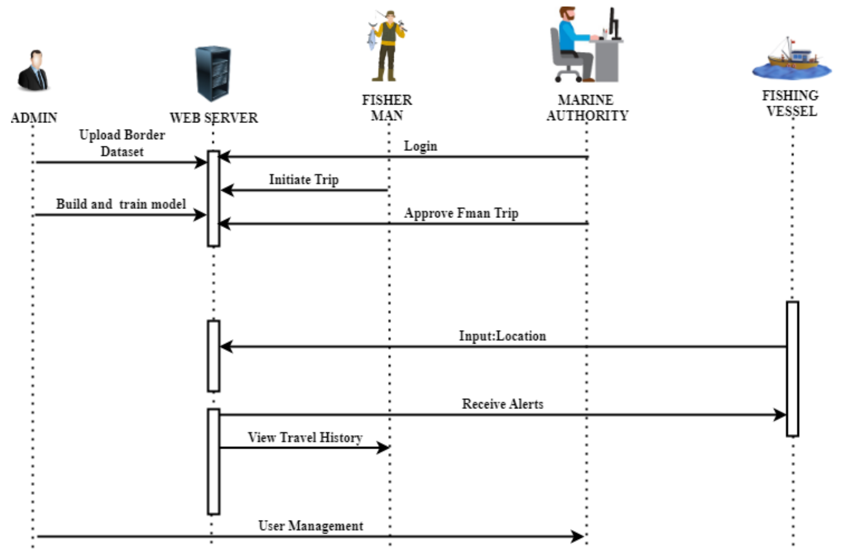
6.5.1. USE CASE DIAGRAM



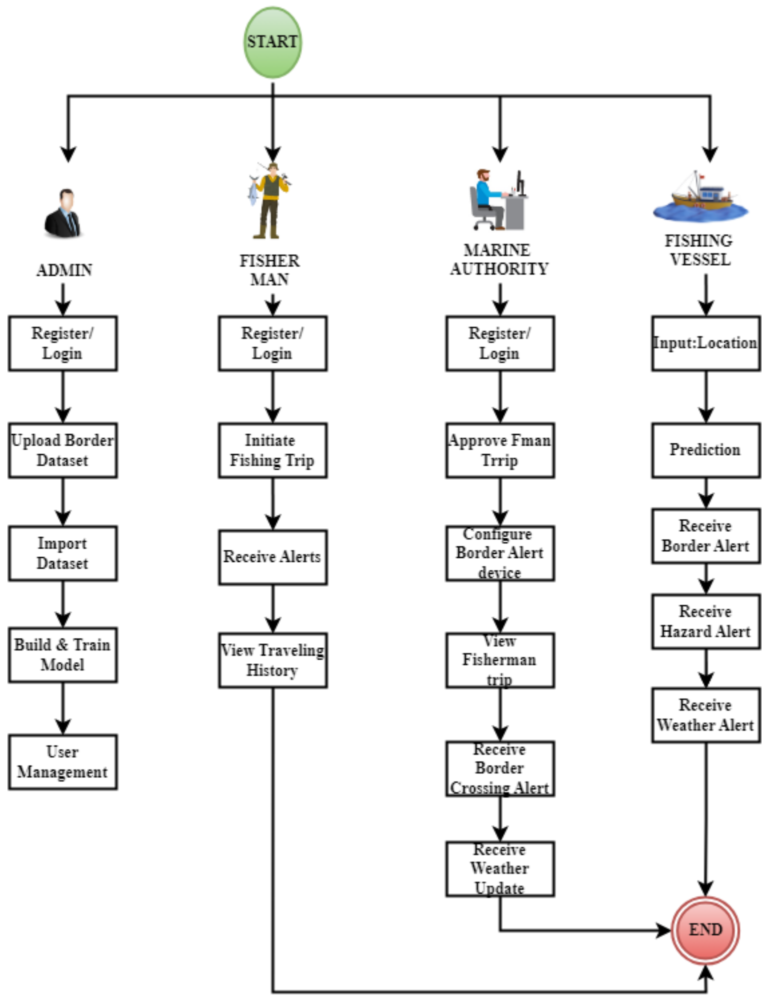
6.5.2. CLASS DIAGRAM



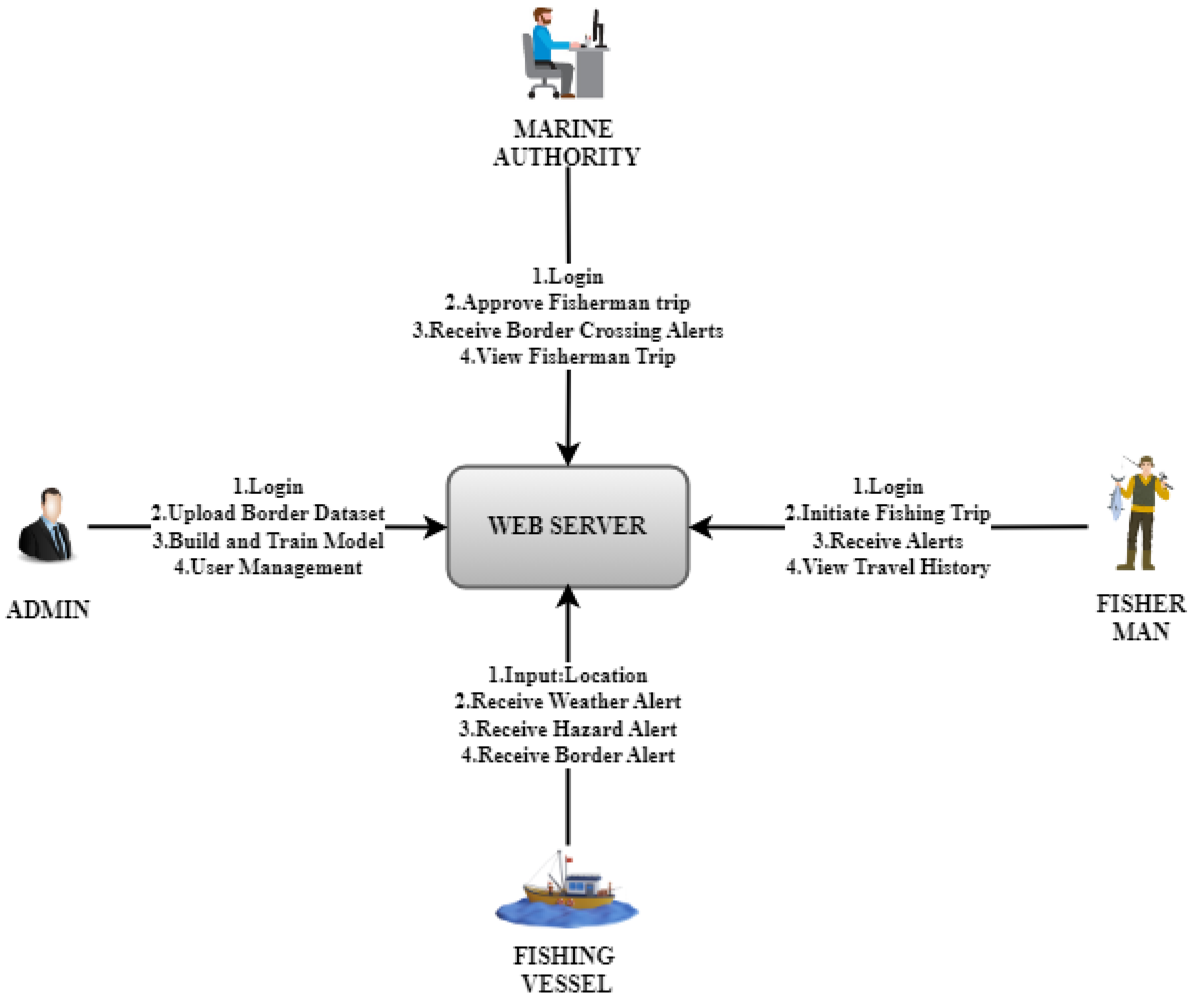
6.5.3. SEQUENCE DIAGRAM



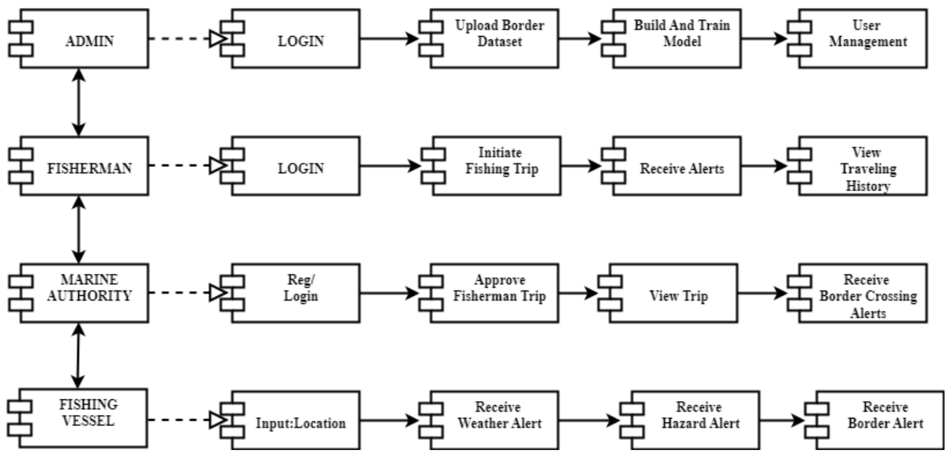
6.5.4. ACTIVITY DIAGRAM



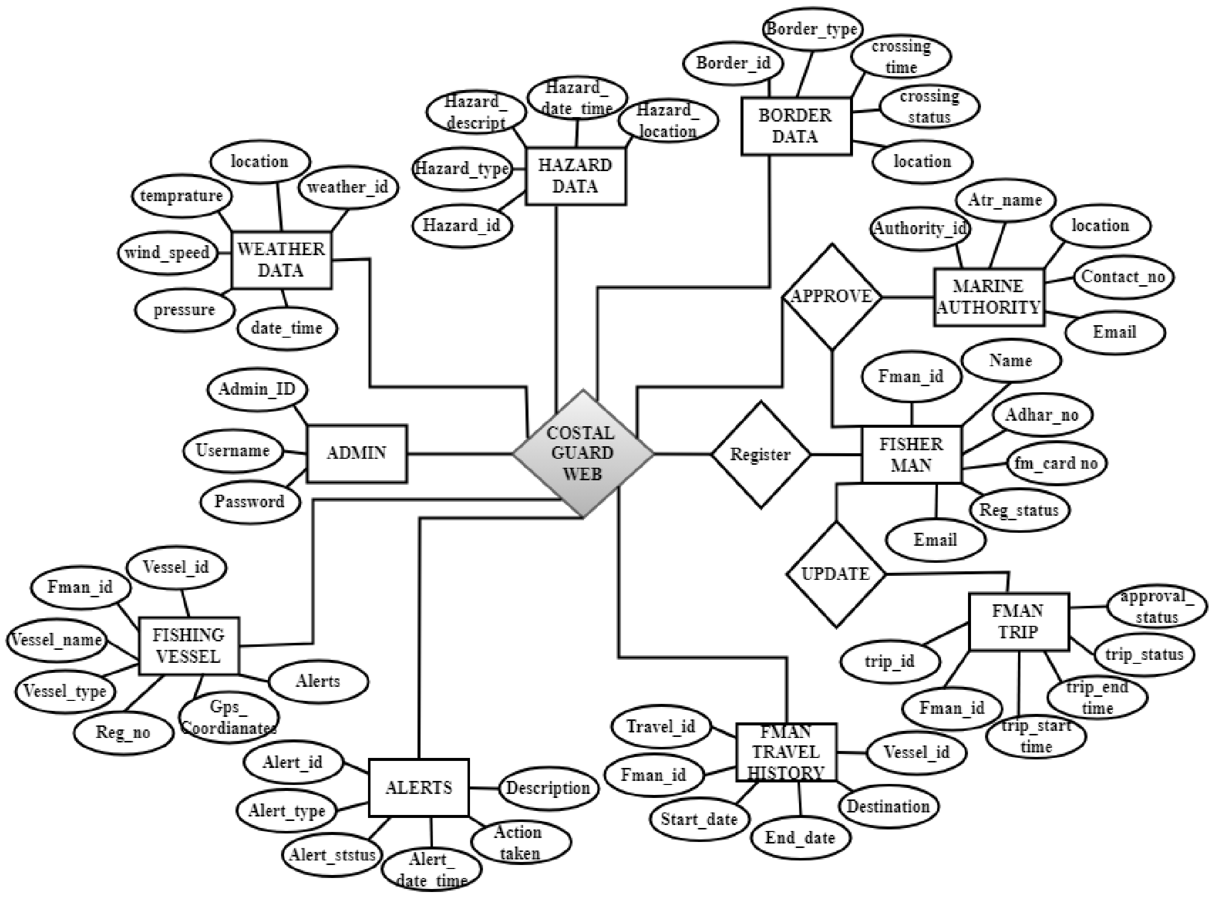
6.5.5. COLLABORATE DIAGRAM



6.5.6. COMPONENT DIAGRAM



6.6. ER DIAGRAM



6.7. DATABASE DESIGN

Web Admin Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Admin_Id	Int	11	Primary Key	Unique Identifier For Each Web Admin.
2	Username	Varchar	50	Not Null	Username For The Web Admin Login.
3	Password	Varchar	100	Not Null	Password For The Web Admin Login.

Fisherman Registration Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Fisherman_Id	Int	11	Primary Key	Unique Identifier For Each Fisherman.
2	Name	Varchar	100	Not Null	Full Name Of The Fisherman.
3	Aadhar Number	Varchar	12	Unique, Not Null	Aadhar Card Number Of The Fisherman.
4	Fisherman Card Number	Varchar	15	Unique, Not Null	Unique Fisherman Card Number.
5	Registration Status	Varchar	50	Not Null	Status Of The Registration (E.G., Approved, Pending).
6	Email	Varchar	100	Not Null, Unique	Email Address For Communication.
7	Phone_Number	Varchar	15	Not Null	Contact Phone Number Of The Fisherman.

Maritime Authority Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Maritime Authority Id	Int	11	Primary Key	Unique Identifier For Each Maritime Authority.
2	Authority Name	Varchar	150	Not Null	Name Of The Maritime Authority.
3	Location	Varchar	100	Not Null	Geographic Location Of The Maritime Authority.
4	Contact_Number	Varchar	15	Not Null	Contact Number Of The Maritime Authority.

5	Email	Varchar	100	Not Null	Email Address For Communication.

Fishing Vessel Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Vessel Id	Int	11	Primary Key	Unique Identifier For Each Vessel.
2	Fisherman Id	Int	11	Foreign Key	Reference To The Fisherman Registering The Vessel.
3	Vessel Name	Varchar	100	Not Null	Name Of The Vessel.
4	Vessel Type	Varchar	50	Not Null	Type Of The Vessel (E.G., Fishing Boat).
5	Registration Number	Varchar	20	Unique, Not Null	Vessel's Unique Registration Number.
6	Gps Coordinates	Varchar	50	Not Null	Real-Time Gps Coordinates Of The Vessel.
7	Alert Status	Varchar	50	Not Null	Status Of Any Border Crossing Alerts.

Fisherman Trip Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Trip_Id	Int	11	Primary Key	Unique Identifier For Each Trip.
2	Fisherman_Id	Int	11	Foreign Key	Id Of The Fisherman.
3	Trip_Start_Time	Datetime	-	Not Null	Start Time Of The Fishing Trip.
4	Trip_End_Time	Datetime	-	Null	End Time Of The Fishing Trip (If Applicable).
5	Trip_Status	Varchar	50	Not Null	Status Of The Trip (E.G., Pending, In-Progress, Completed).
6	Approval_Status	Varchar	50	Not Null	Approval Status Of The Trip (Approved/Denied/Pending).

Weather Data Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Weather_Id	Int	11	Primary Key	Unique Identifier For Weather Updates.
2	Location	Varchar	100	Not Null	Location Of The Weather Data.
3	Temperature	Float	5,2	Not Null	Temperature At The Location (In Celsius).
4	Humidity	Float	5,2	Not Null	Humidity Percentage At The Location.
5	Wind_Speed	Float	5,2	Not Null	Wind Speed At The Location (In M/S).
6	Pressure	Float	5,2	Not Null	Atmospheric Pressure At The Location.
7	Date_Time	Datetime	-	Not Null	Date And Time When Weather Data Was Recorded.

Hazardous Dataset Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Hazard_Id	Int	11	Primary Key	Unique Identifier For Each Hazard Entry.
2	Hazard_Type	Varchar	100	Not Null	Type Of Hazard (E.G., Storm, Navigation Error).
3	Hazard_Description	Text	-	Not Null	Description Of The Hazardous Event.
4	Hazard_Severity	Varchar	50	Not Null	Severity Of The Hazard (E.G., Low, Medium, High).
5	Hazard_Date_Time	Datetime	-	Not Null	Timestamp When The Hazard Occurred.
6	Hazard_Location	Varchar	150	Not Null	Location Of The Hazard (Gps Coordinates Or Region).

Border Dataset Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Border_Id	Int	11	Primary Key	Unique Identifier For Each Border Event.
2	Border_Type	Varchar	100	Not Null	Type Of Border Crossing Event (E.G., Legal, Illegal).

3	Crossing_Status	Varchar	50	Not Null	Status Of The Crossing (E.G., Approved, Denied, Pending).
4	Border_Cross_Time	Datetime	-	Not Null	Timestamp Of When The Border Was Crossed.
5	Border_Location	Varchar	150	Not Null	Location Of The Border Crossing (E.G., Gps Coordinates).

Alerts Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Alert_Id	Int	11	Primary Key	Unique Identifier For Each Alert.
2	Alert_Type	Varchar	50	Not Null	Type Of Alert (E.G., Weather Alert, Border Crossing Alert, Hazardous Alert).
3	Alert_Status	Varchar	50	Not Null	Current Status Of The Alert (E.G., Triggered, Resolved, Pending).
4	Alert_Date_Time	Datetime	-	Not Null	Date And Time When The Alert Was Generated.
5	Action_Taken	Text	-		Action Taken In Response To The Alert.
6	Description	Text	-		Additional Description About The Alert, Including Severity Or Specific Details.

Fisherman Travel History Table

S.No	Field	Data Type	Field Size	Constraint	Description
1	Travel_Id	Int	11	Primary Key	Unique Identifier For Each Fishing Trip.
2	Fisherman_Id	Int	11	Foreign Key	Reference To The Fisherman.
3	Start_Date	Datetime	-	Not Null	Start Date And Time Of The Trip.
4	End_Date	Datetime	-		End Date And Time Of The Trip.
5	Destination	Varchar	100		Destination Of The Fishing Trip.
6	Vessel_Id	Int	11	Foreign Key	Reference To The Vessel Used For The Trip.

CHAPTER 7
SYSTEM IMPLEMENTATION

The implementation of the proposed system involves several key components and steps:

1. **Backend Development**
    - Implementation of backend functionalities using Python and Flask framework.
    - Setup and configuration of MySQL database for data storage and retrieval.
2. **Frontend Development**
    - Design and development of user interfaces using HTML, CSS, and JavaScript.
    - Utilization of Bootstrap framework for responsive and mobile-friendly designs.
3. **Database Management**
    - Creation of database schema to store user profiles, border surveillance data, alerts, and other relevant information.
    - Implementation of database queries for CRUD operations (Create, Read, Update, Delete) to manage data effectively.
4. **Deep Learning Model Development**
    - Collection and preprocessing of historical border surveillance data for training the BorderNet model.
    - Design and implementation of LSTM neural network architecture for border breach prediction.
5. **Integration of External APIs**
    - Integration of weather data APIs to fetch real-time meteorological information for weather alerts and forecasts.
    - Incorporation of GPS tracking APIs to receive continuous location updates from fishing vessels for border surveillance.
6. **Alert Notification System:**
    - Development of alert notification mechanisms using SMS, email, and push notifications for timely dissemination of information.
    - Configuration of notification triggers based on predefined thresholds and conditions for different types of alerts.
7. **User Authentication and Access Control:**

- Implementation of user authentication mechanisms to secure access to system functionalities.
- Assignment of user roles and permissions to control access to sensitive data and features.

8. **Testing and Quality Assurance:**
    - Conducting unit tests, integration tests, and system tests to validate the functionality and reliability of the system.
    - Identification and resolution of bugs, errors, and performance bottlenecks through iterative testing and debugging processes.
9. **Deployment and Hosting:**
    - Deployment of the system on a web server.
    - Configuration of server environments and network settings for optimal performance and scalability.

7.1. SYSTEM DESCRIPTION

The project is developed using Python, Flask, MySQL, and Wampserver for backend infrastructure and local server hosting. Machine learning capabilities, facilitated by TensorFlow and Keras, enable predictive analytics, while Pandas, NumPy, Matplotlib, and Seaborn handle data processing and visualization. Bootstrap ensures a responsive frontend for coast guard personnel, enhancing usability. This centralized platform facilitates real-time border surveillance, communication with authorities, and coordination with fishermen. Immediate alerts for breaches, weather changes, and hazards, along with reporting tools and user management, bolster maritime security.

7.2. MODULES DESCRIPTION

1. Coast Guard Stations Web App

The Coast Guard Stations Web App is being developed using Python and Flask for the backend, MySQL for database management, and Wampserver for local server hosting. Integration of machine learning capabilities is achieved with TensorFlow and Keras for predictive analytics, supported by Pandas, Scikit Learn, NumPy, Matplotlib, and Seaborn for data processing, visualization, and model training.

2. BorderNet Model: Build and Train

The BorderNet Model: Build and Train Module encompasses several key steps in developing a robust predictive model for classifying borders as either safe or breached. Here's a detailed description of each component:

2.1. Dataset Description

The dataset for the Tamil Nadu to Sri Lanka border consists of various features and labels aimed at training the Bordernet model to classify borders as safe or breached. Here's an example of what the dataset might look like:

Features

- **Timestamp**: Date and time of the border observation.
- **Latitude**: Geographic latitude of the border location.
- **Longitude**: Geographic longitude of the border location.
- **Speed**: Speed of vessels in the vicinity of the border.
- **Direction**: Direction of vessel movement relative to the border.
- **Weather Conditions**: Meteorological data such as wind speed, visibility, and precipitation.
- **Temperature**: Ambient temperature at the border location.
- **Sea State**: Conditions of the sea surface, including wave height and swell.
- **Marine Traffic**: Density and type of maritime traffic in the area.

Labels:

- **Border Status**: Classification of the border as either "safe" or "breached" based on observed activity and security assessments.

Example Entry

Timestamp	**Lat**	**Lon**	**Spd**	**Dir**	**WC**	**Temp**	**SS**	**MT**	**BS**
2023-05-15 10:00:00	9.850	79.976	12	NE	Clear	28°C	Moderate	Moderate	Safe
2023-05-15 11:00:00	9.845	79.972	8	N	Partly Cloudy	29°C	Moderate	Light	Breached
2023-05-15 12:00:00	9.840	79.968	10	NW	Clear	30°C	Moderate	Heavy	Safe

In this example, each row represents an observation of the border at a specific timestamp. Features such as latitude, longitude, speed, direction, weather conditions, temperature, sea state, and marine traffic are recorded. The "Border Status" column indicates whether the border was classified as "safe" or "breached" based on observed activity.

2.2. Import Dataset

The Import Dataset module serves to gather, organize, and load the dataset required for training and testing the BorderNet model. Its primary function is to import data from various sources and format it into a structured format suitable for model training. It is implemented using Python libraries such as Pandas, NumPy, and Matplotlib for data manipulation and visualization. It utilizes functions and classes to automate data loading and visualization tasks, streamlining the dataset preparation process.

2.3. Preprocessing

The Preprocessing Module aims to ensure that the dataset used for training the BorderNet model is clean, consistent, and suitable for analysis. It focuses on handling null values, redundant data, and missing values, as well as performing data cleaning and removing duplicate entries. This module is implemented using Python libraries such as Pandas and NumPy for data manipulation

- **Handling Null Values**

The module identifies and handles null values within the dataset using appropriate techniques such as imputation or deletion. Imputation methods like mean, median, or mode replacement may be used to fill missing values in numerical features, while categorical features may be imputed with the most frequent value.

- **Handling Redundant Data**

Redundant data, which includes duplicate observations or features with identical information, is identified and removed from the dataset. This ensures that each observation contributes unique information to the model training process, preventing bias and overfitting.

- **Handling Missing Values**

Missing values in the dataset are addressed through imputation or deletion, depending on the nature and extent of the missing data. Techniques such as mean, median, or mode imputation may be used for numerical features, while categorical features may be imputed with the most frequent category.

- **Data Cleaning**

Data cleaning involves identifying and correcting errors, inconsistencies, or outliers in the dataset. This may include removing outliers that skew the distribution of data, correcting data entry errors, and standardizing data formats to ensure consistency across features.

Output

The output of the Preprocessing Module is a cleaned and standardized dataset ready for model training. This dataset is free from null values, redundant data, and missing values, and has undergone data cleaning to address errors and inconsistencies. It serves as the input to the model training process, ensuring that the BorderNet model is trained on high-quality data.

2.4. Feature Extraction

The Feature Extraction Module aims to identify and select the most informative features that contribute to the prediction task of classifying borders as safe or breached. It leverages a correlation matrix to assess the relationships between features and target variables, guiding the selection of relevant features for model training. It is implemented using Python libraries such as Pandas, NumPy, and Seaborn for data manipulation, correlation analysis, and visualization. Custom functions are utilized to compute the correlation matrix and extract informative features based on correlation coefficients.

- **Correlation Matrix Computation**:

The module computes a correlation matrix, which is a square matrix containing correlation coefficients between pairs of features in the dataset. This matrix provides insight into the linear relationships between features, indicating how changes in one variable may affect changes in another.

- **Assessment of Feature Relevance**:

The correlation matrix is analyzed to assess the strength and direction of correlations between features and the target variable (border status). Features with high correlation coefficients (either positive or negative) with the target variable are considered more relevant and informative for model training.

- **Selection of Informative Features**:

Based on the correlation matrix analysis, features with significant correlations with the target variable are selected for inclusion in the model training dataset. This selection process ensures that only the most relevant features are used to train the BorderNet model, improving its predictive performance.

2.5. Classification

The Classification Module aims to implement a classification algorithm, specifically LSTM neural networks, to classify borders into two categories: safe or breached. By leveraging the temporal dependencies inherent in border surveillance data, LSTM networks can effectively capture complex patterns and dynamics, making them well-suited for this task.

LSTM Network Architecture

The LSTM network architecture defines the layout and configuration of the neural network, including the number of layers, hidden units, activation functions, and other parameters. It plays a crucial role in determining the network's capacity to capture temporal dependencies and classify borders accurately.

Key Components:

- **Number of Layers**

The LSTM network may consist of one or more layers of LSTM cells stacked on top of each other. Deeper networks with multiple layers can capture more complex patterns and relationships in the data but may require longer training times and are more prone to overfitting.

- **Hidden Units**

Each LSTM cell in the network contains a set of hidden units, also known as neurons or memory cells. The number of hidden units determines the capacity of the network to learn and represent complex patterns in the data.

- **Activation Functions**

Activation functions introduce non-linearity into the network, allowing it to model complex relationships between input features and output labels. Common activation functions used in LSTM networks include the sigmoid function for gating mechanisms and the hyperbolic tangent (tanh) function for output activations.

- **Dropout**

Dropout is a regularization technique used to prevent overfitting by randomly dropping a fraction of connections between LSTM cells during training. It helps improve the generalization performance of the network by reducing reliance on specific features or connections.

- **Recurrent Connections**

LSTM networks contain recurrent connections that allow information to flow across time steps, enabling the network to process sequential data and capture long-term dependencies. These connections facilitate the retention and utilization of historical information, enhancing the network's predictive capabilities.

2.6. Build and Train

Build and Train Module aims to create and train the LSTM neural network, which serves as the core component of the Bordernet system. The module defines the architecture of the LSTM network, specifies hyperparameters, and optimizes model performance through iterative training and validation processes.

- **Hyperparameter Specification**

Hyperparameters such as learning rate, batch size, number of epochs, and optimizer choice are specified to configure the training process. These hyperparameters influence the convergence and generalization performance of the model and are optimized through experimentation and validation.

- **Model Compilation**

The LSTM model is compiled with the specified architecture and hyperparameters, configuring the optimization algorithm, loss function, and evaluation metrics. This step prepares the model for training by specifying how it should update its parameters during the training process.

- **Training and Validation**

The preprocessed dataset is split into training and validation subsets, with the training data used to update the model's parameters and the validation data used to assess its performance on unseen examples. The model is trained iteratively over multiple epochs, with performance monitored and evaluated at each epoch.

2.7. Deploy Model

The Deploy Model Module aims to seamlessly integrate the trained LSTM model into the Coast Guard Stations Web App, allowing coast guard personnel to access its predictive capabilities for real-time border classification. This integration empowers personnel with actionable insights to enhance border security operations.

3. Border Crossing Prediction

The Border Crossing Prediction Module serves as a proactive tool in maritime security efforts, anticipating potential border crossings by fishing vessels through a combination of real-time location tracking and predictive analytics. Here's a detailed description of each component within this module.

3.1. Fishing Vessels Sailing Animation:

This component provides a dynamic visualization of fishing vessels' movements through a sailing animation interface. It displays real-time updates of vessel positions and trajectories, allowing coast guard personnel to monitor maritime activity visually. The animation provides an intuitive understanding of vessel movements, facilitating rapid detection of suspicious behavior or deviations from expected routes.

3.2. Continuously Input Location

Integrated BorderAlert GPS devices are installed on fishing vessels to continuously transmit location data to the Border Crossing Prediction Module. These devices provide accurate and up-to-date information on vessel positions, enabling the module to track vessel movements in real time.

3.3. Predict Border Crossing

The predictive capability of the module relies on the trained BorderNet Model, which utilizes LSTM neural networks to forecast potential border crossings by fishing vessels. By analyzing historical data on vessel trajectories, maritime boundaries, and border activity, the model identifies patterns and trends indicative of potential breaches.

4. Alert System

The Alert System Module is designed to provide timely and relevant alerts to fishermen and maritime authorities, helping them mitigate risks and ensure safety in maritime environments. It plays a crucial role in enhancing safety and security in maritime environments by providing timely alerts to fishermen and maritime authorities regarding potential risks and hazards.

4.1. Border Alert

This component issues alerts to both fishermen and maritime authorities in the event of detected border breaches or suspicious activity. Alerts may include notifications via mobile applications, SMS, or email, informing fishermen of potential security threats and prompting them to take appropriate action. Simultaneously, maritime authorities receive alerts to coordinate response efforts, deploy resources, and investigate potential security breaches.

4.2. Weather Alert

The Weather Alert component notifies fishermen of adverse weather conditions that may pose risks to their safety or affect fishing operations. Alerts are issued based on real-time meteorological data, providing information on factors such as strong winds, heavy rainfall, thunderstorms, or rough seas.

4.3. Hazard Alert

The Hazard Alert component warns fishermen about potential hazards or dangers in their vicinity, such as submerged rocks, navigational hazards, or other vessels in distress. Alerts are generated based on data from maritime surveillance systems, AIS (Automatic Identification System), or crowd-sourced reports from other vessels.

5. System User

The System User Module facilitates interaction between various stakeholders, including maritime authorities, fishermen, and weather data providers, within the maritime security system. Here's a detailed description of each user category and their respective functionalities:

5.1. Maritime Authorities

Login: Maritime authorities can access the system by logging in with their credentials, ensuring secure access to sensitive information and functionalities.

Approve Fisherman Registration Request: Authorities review and approve registration requests from fishermen, validating their identity and credentials before granting access to the system.

Configure BorderAlert Device to Fisherman Vessel: Authorities configure BorderAlert devices on fishermen vessels, ensuring seamless integration with the surveillance system for real-time border monitoring.

Receive Border Crossing Alert of Fisherman: Authorities receive instant alerts in case of detected border crossings or suspicious activities by fishermen, enabling prompt response and intervention.

View Weather: Authorities can access real-time weather information through the system, allowing them to assess environmental conditions and make informed decisions regarding maritime operations.

View Fisherman Travel Details: Authorities have access to detailed information about fishermen's travel history, including routes taken, duration of trips, and areas visited, facilitating oversight and monitoring of maritime activities.

5.2. Fisherman

Register with Aadhar and Fisherman Card: Fishermen register on the system by providing identification details such as Aadhar and Fisherman Card, verifying their identity and credentials.

Receive Registration Approval: Upon registration, fishermen await approval from maritime authorities, ensuring compliance with regulations and access to system functionalities.

Login: Approved fishermen access the system by logging in with their credentials, enabling them to utilize various features and services.

Update Profile: Fishermen can update their profile information, including contact details, vessel information, and emergency contacts, ensuring accurate and up-to-date records.

Initiate Fishing Travel with Date and Time: Fishermen initiate fishing trips by specifying the date, time, and intended route, enabling authorities to track their movements and ensure safety.

Receive Alerts: Fishermen receive alerts in real-time, including border crossing alerts, weather updates, and hazard warnings, enhancing situational awareness and safety at sea.

View Travel History: Fishermen can access their travel history, including past trips, routes taken, and duration of journeys, providing insights into their maritime activities.

5.3. Weather Data Provider API

Update Weather Information to the Coast Guard Stations Web App: The weather data provider API updates real-time weather information to the Coast Guard Stations Web App, ensuring accurate and up-to-date meteorological data for informed decision-making by authorities and fishermen.

CHAPTER 8
SYSTEM TESTING

System testing of the project involved validating the functionality, performance, security, and usability of the entire system as a whole. Here's an overview of the testing process:

1. **Functional Testing**
   - Verified that all features and functionalities outlined in the system requirements worked correctly.
2. **Performance Testing**
   - Assessed the responsiveness and speed of the web application, especially during peak usage times.
3. **Security Testing**
   - Conducted penetration testing to identify and address vulnerabilities in the system, including SQL injection, cross-site scripting, and unauthorized access.
4. **Usability Testing**
   - Gathered feedback from users to assess the ease of use, intuitiveness, and overall user experience of the system.
5. **Compatibility Testing**
   - Tested the compatibility of the web application with various web browsers (e.g., Chrome, Firefox, Safari) and operating systems (e.g., Windows, macOS, Linux).
6. **Regression Testing**
   - Performed regression testing to ensure that recent updates or changes to the system had not introduced any new bugs or issues.
7. **End-to-End Testing**
   - Conducted end-to-end testing to validate the entire workflow of the system, from data input and processing to alert generation and user interaction.

8.1. TEST CASES

Test Cases for Maritime Authorities

1. **Test Case ID**: FBS_MA_TC_001
   - **Input**: Maritime authority logs in with valid credentials.

- **Expected Result**: Authority should be successfully logged in and redirected to the dashboard.
- **Actual Result**: Authority was logged in successfully, and the dashboard was displayed.
- **Status**: Pass

2. **Test Case ID**: FBS_MA_TC_002
    - **Input**: Maritime authority approves a fisherman registration request.
    - **Expected Result**: Fisherman's registration status should be updated to approved, and a confirmation message should be sent.
    - **Actual Result**: Fisherman's registration status was updated to approved, and a confirmation email was sent.
    - **Status**: Pass
3. **Test Case ID**: FBS_MA_TC_003
    - **Input**: Maritime authority configures BorderAlert device to a fisherman vessel.
    - **Expected Result**: Device should be successfully configured and linked to the fisherman's vessel for border monitoring.
    - **Actual Result**: BorderAlert device was successfully configured and linked to the fisherman's vessel.
    - **Status**: Pass

Test Cases for Fishermen

1. **Test Case ID**: FBS_F_TC_001
    - **Input**: Fisherman registers with Aadhar and Fisherman Card.
    - **Expected Result**: Fisherman's registration should be successfully submitted for approval.
    - **Actual Result**: Fisherman's registration was submitted successfully, awaiting approval.
    - **Status**: Pass
2. **Test Case ID**: FBS_F_TC_002
    - **Input**: Fisherman receives registration approval.
    - **Expected Result**: Fisherman should receive confirmation of registration approval.

- **Actual Result**: Fisherman received a confirmation email "Your registration has been approved."
- **Status**: Pass

3. **Test Case ID**: FBS_F_TC_003
    - **Input**: Fisherman logs in with valid credentials.
    - **Expected Result**: Fisherman should be successfully logged in and redirected to the dashboard.
    - **Actual Result**: Fisherman was logged in successfully, and the dashboard was displayed.
    - **Status**: Pass
4. **Test Case ID**: FBS_F_TC_004
    - **Input**: Fisherman updates profile information.
    - **Expected Result**: Fisherman's profile should be successfully updated with the new information.
    - **Actual Result**: Fisherman's profile was updated successfully with the new contact details.
    - **Status**: Pass

8.2. TEST REPORT

Introduction

The project is undergoing testing to ensure its functionality, usability, and reliability in enhancing maritime security operations. This test report outlines the test objectives, scope, environment, results, and bug reports identified during testing.

Test Objective

The objective of this testing is to validate the functionality and performance of the Coast Guard Stations Web App, including features such as user authentication, border monitoring, alert systems, and integration with external data sources.

Test Scope

The testing scope includes functional testing of key features, usability testing to assess user interaction, and performance testing to evaluate system responsiveness and reliability under various conditions.

Test Environment

- **Backend Infrastructure**: Python, Flask, MySQL, Wampserver
- **Machine Learning Frameworks**: TensorFlow, Keras
- **Data Processing & Visualization**: Pandas, NumPy, Matplotlib, Seaborn
- **Frontend Framework**: Bootstrap
- **Testing Tools**: Selenium, JUnit
- **Operating System**: Windows 10
- **Browsers**: Chrome, Firefox, Edge

Test Result

The testing results indicate that the Coast Guard Stations Web App performed satisfactorily across various test scenarios, with the majority of test cases passing successfully. However, a few bugs were identified and reported during testing.

Bug Report

A bug report identifies any deviations from expected behavior observed during testing. Each bug is assigned a unique ID and linked to the corresponding test case where the issue occurred.

BID	TCID	Bug Description	Bug Status	Output
B001	FBS_MA_TC_004	Maritime authority does not receive alert	Fixed	Authority did not receive border alert email
B002	FBS_F_TC_006	Fisherman receives duplicate alert	Open	Fisherman received multiple border alerts

Test Conclusion

Overall, the project demonstrated robust functionality and usability, meeting the intended objectives of enhancing maritime security operations. The identified bugs have been documented and will be addressed in subsequent development iterations to ensure optimal performance and user experience.

CHAPTER 9
BOTTOM LINE

9.1. CONCLUSION

In conclusion, this project fortified with the BorderNet model and advanced alert systems, emerges as a robust solution for bolstering maritime security operations. Through the amalgamation of cutting-edge technologies like machine learning, real-time data integration, and predictive analytics, this project significantly elevates the efficacy of border surveillance and response mechanisms. The user-centric design of the system, coupled with its ability to customize alert preferences, ensures accessibility and usability for coast guard personnel and fishermen. By providing continuous monitoring of maritime borders and issuing timely alerts for potential breaches, adverse weather conditions, and maritime hazards, the system plays a pivotal role in enhancing safety and security at sea. Moreover, the feasibility analysis underscores the project's practicality and potential for real-world implementation.

9.2. FUTURE ENHANCEMENT

- **Integration of Additional Data Sources:** Incorporating data from satellite imagery, unmanned aerial vehicles (UAVs), and other sensor networks can provide more comprehensive situational awareness, enabling better-informed decision-making.
- **Geofencing Capabilities:** Introducing geofencing capabilities to define virtual boundaries and trigger alerts when vessels enter or exit designated areas can enhance targeted surveillance and response efforts.
- **Mobile Application Development:** Developing dedicated mobile applications for coast guard personnel and fishermen can extend the reach of the system, enabling real-time access to alerts, status updates, and reporting functionalities from anywhere.
- **Incident Response Automation:** Implementing automated incident response workflows that leverage artificial intelligence (AI) and natural language processing (NLP) can streamline response efforts and ensure rapid coordination between stakeholders in emergency situations.

APPENDIX

SOURCE CODE

Packages

```
from flask import Flask, render_template, Response, redirect, request, session, abort, url_for
import os
import base64
import cv2
import pandas as pd
import numpy as np
import imutils
from flask import send_file
from werkzeug.utils import secure_filename
import shutil
import json
import csv
import subprocess
import mysql.connector
import hashlib
from datetime import datetime
from datetime import date
import datetime
import time
from random import seed
from random import randint
from PIL import Image
import stepic
import urllib.parse
from urllib.request import urlopen
import webbrowser
```

Database Connection

```
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
charset="utf8",
database="fisherman"
```

Login

```
def login_fm():
msg=""
if request.method=='POST':
uname=request.form['uname']
pwd=request.form['pass']
cursor = mydb.cursor()
cursor.execute('SELECT * FROM fm_fisher WHERE uname = %s AND pass = %s', (uname,
pwd))
account = cursor.fetchone()
if account:
session['username'] = uname
return redirect(url_for('fm_home'))
else:
msg = 'Incorrect username/password!'
```

Fisherman Registration

```
def register():
msg=""
fn=""
email=""
mess=""
act=request.args.get("act")
mycursor = mydb.cursor()
mycursor.execute("SELECT max(id)+1 FROM fm_fisher")
maxid = mycursor.fetchone()[0]
if maxid is None:
```

```
maxid=1
now = datetime.datetime.now()
rdate=now.strftime("%d-%m-%Y")
if request.method=='POST':
name=request.form['name']
gender=request.form['gender']
dob=request.form['dob']
address=request.form['address']
mobile=request.form['mobile']
email=request.form['email']
pass1=request.form['pass']
file = request.files['file']
file2 = request.files['file2']
uname="F"+str(maxid)
filename=file.filename
aadhar="A"+str(maxid)+filename
file.save(os.path.join("static/upload", aadhar))
filename2=file2.filename
fcard="F"+str(maxid)+filename2
file2.save(os.path.join("static/upload", fcard))
mess="Dear "+name+", Fisherman ID:"+uname+", Password:"+pass1
sql = "INSERT INTO
fm_fisher(id,name,gender,dob,address,mobile,email,aadhar,fisher_card,uname,pass,rdate,approv
e_st) VALUES (%s,%s,%s, %s, %s, %s, %s, %s, %s, %s, %s, %s,%s)"
val = (maxid,name,gender,dob,address,mobile,email,aadhar,fcard,uname,pass1,rdate,'0')
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "Registered Success")
msg="success"
```

Get Weather Data

```
lat1=lat[0:8]
```

```
lon1=lon[0:8]
responsetext="https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timel
ine/"+loc+"/"+sdate+"/"+edate+"?unitGroup=metric&include=days&key=8STCAMRSRTEZ77J
A2XRP7FMNC&contentType=csv"
cdata=[]
data1 = pd.read_csv(responsetext, header=0)
for ss in data1.values:
dt=[]
with open("static/weather_data.csv",'a',newline='') as outfile:
writer = csv.writer(outfile, quoting=csv.QUOTE_NONNUMERIC)
writer.writerow(ss)
cdata.append(dt)
Training
def process1():
uname=""
msg=""
path='dataset/'
df=pd.read_csv('static/indian-ocean-data.csv')
dat1=df.head(100)
data1=[]
for ss1 in dat1.values:
data1.append(ss1)
#LSTM
def load_data(stock, seq_len):
amount_of_features = len(stock.columns)
data = stock.as_matrix() #pd.DataFrame(stock)
sequence_length = seq_len + 1
result = []
for index in range(len(data) - sequence_length):
result.append(data[index: index + sequence_length])
```

```
result = np.array(result)
row = round(0.9 * result.shape[0])
train = result[:int(row), :]
x_train = train[:, :-1]
y_train = train[:, -1][:,-1]
x_test = result[int(row):, :-1]
y_test = result[int(row):, -1][:,-1]
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], amount_of_features))
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], amount_of_features))
return [x_train, y_train, x_test, y_test]
def train(train_iter, dev_iter, test_iter, model, args):
if args.cuda:
model.cuda()
if args.Adam is True:
print("Adam Training......")
optimizer = torch.optim.Adam(model.parameters(), lr=args.lr,
weight_decay=args.init_weight_decay)
elif args.SGD is True:
print("SGD Training.......")
optimizer = torch.optim.SGD(model.parameters(), lr=args.lr,
weight_decay=args.init_weight_decay,
momentum=args.momentum_value)
elif args.Adadelta is True:
print("Adadelta Training.......")
optimizer = torch.optim.Adadelta(model.parameters(), lr=args.lr,
weight_decay=args.init_weight_decay)
steps = 0
model_count = 0
best_accuracy = Best_Result()
model.train()
for epoch in range(1, args.epochs+1):
```

```
steps = 0
print("\n## The {} Epoch, All {} Epochs ! ##".format(epoch, args.epochs))
for batch in train_iter:
feature, target = batch.text, batch.label.data.sub_(1)
if args.cuda is True:
feature, target = feature.cuda(), target.cuda()
target = autograd.Variable(target)  # question 1
optimizer.zero_grad()
logit = model(feature)
loss = F.cross_entropy(logit, target)
loss.backward()
if args.init_clip_max_norm is not None:
utils.clip_grad_norm_(model.parameters(), max_norm=args.init_clip_max_norm)
optimizer.step()
steps += 1
if steps % args.log_interval == 0:
train_size = len(train_iter.dataset)
corrects = (torch.max(logit, 1)[1].view(target.size()).data == target.data).sum()
accuracy = float(corrects)/batch.batch_size * 100.0
sys.stdout.write(
'\rBatch[{}/{}] - loss: {:.6f}  acc: {:.4f}%({}/{})'.format(steps,
train_size,
loss.item(),
accuracy,
corrects,
batch.batch_size))
if steps % args.test_interval == 0:
print("\nDev  Accuracy: ", end="")
eval(dev_iter, model, args, best_accuracy, epoch, test=False)
print("Test Accuracy: ", end="")
eval(test_iter, model, args, best_accuracy, epoch, test=True)
```

```
if steps % args.save_interval == 0:
if not os.path.isdir(args.save_dir): os.makedirs(args.save_dir)
save_prefix = os.path.join(args.save_dir, 'snapshot')
save_path = '{}_steps{}.pt'.format(save_prefix, steps)
torch.save(model.state_dict(), save_path)
if os.path.isfile(save_path) and args.rm_model is True:
os.remove(save_path)
model_count += 1
return model_count
def eval(data_iter, model, args, best_accuracy, epoch, test=False):
model.eval()
corrects, avg_loss = 0, 0
for batch in data_iter:
feature, target = batch.text, batch.label
target.data.sub_(1)
if args.cuda is True:
feature, target = feature.cuda(), target.cuda()
logit = model(feature)
loss = F.cross_entropy(logit, target)
avg_loss += loss.item()
corrects += (torch.max(logit, 1)[1].view(target.size()).data == target.data).sum()
size = len(data_iter.dataset)
avg_loss = loss.item()/size
accuracy = float(corrects)/size * 100.0
model.train()
print(' Evaluation - loss: {:.6f}  acc: {:.4f}%({}/{}) '.format(avg_loss, accuracy, corrects, size))
if test is False:
if accuracy >= best_accuracy.best_dev_accuracy:
best_accuracy.best_dev_accuracy = accuracy
best_accuracy.best_epoch = epoch
best_accuracy.best_test = True
```

```
if test is True and best_accuracy.best_test is True:
best_accuracy.accuracy = accuracy
if test is True:
print("The Current Best Dev Accuracy: {:.4f}, and Test Accuracy is :{:.4f}, locate on {}
epoch.\n".format(
best_accuracy.best_dev_accuracy, best_accuracy.accuracy, best_accuracy.best_epoch))
if test is True:
best_accuracy.best_test = False
def build_model(layers):
model = Sequential()
model.add(LSTM(
input_dim=layers[0],
output_dim=layers[1],
return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(
layers[2],
return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(
output_dim=layers[2]))
model.add(Activation("linear"))
start = time.time()
model.compile(loss="mse", optimizer="rmsprop",metrics=['accuracy'])
print("Compilation Time : ", time.time() - start)
return model
def build_model2(layers):
d = 0.2
model = Sequential()
model.add(LSTM(128, input_shape=(layers[1], layers[0]), return_sequences=True))
model.add(Dropout(d))
model.add(LSTM(64, input_shape=(layers[1], layers[0]), return_sequences=False))
model.add(Dropout(d))
model.add(Dense(16,init='uniform',activation='relu'))
model.add(Dense(1,init='uniform',activation='linear'))
model.compile(loss='mse',optimizer='adam',metrics=['accuracy'])
return model
```

Fisherman Approval

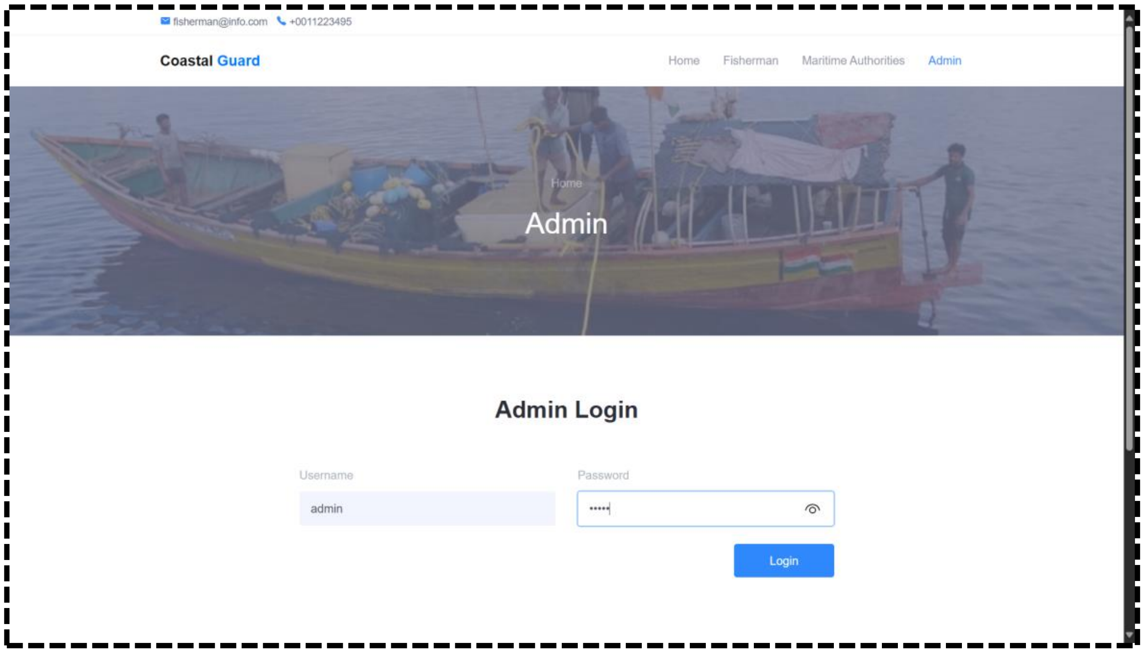
```
def auth_home():
msg=""
uname=""
data4=[]
act=request.args.get("act")
#if 'username' in session:
#    uname = session['username']
ff=open("static/auth.txt","r")
uname=ff.read()
ff.close()
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM fm_authority where uname=%s",(uname,))
data = mycursor.fetchone()
mycursor.execute("SELECT * FROM fm_fisher")
data1 = mycursor.fetchall()
if act=="ok":
fid=request.args.get("fid")
mycursor.execute("update fm_fisher set approve_st=1 where id=%s",(fid,))
mydb.commit()
msg="yes"
data11 = pd.read_csv("static/weather_data.csv", header=0)
r=0
for ss in data11.values:
dt=[]
```

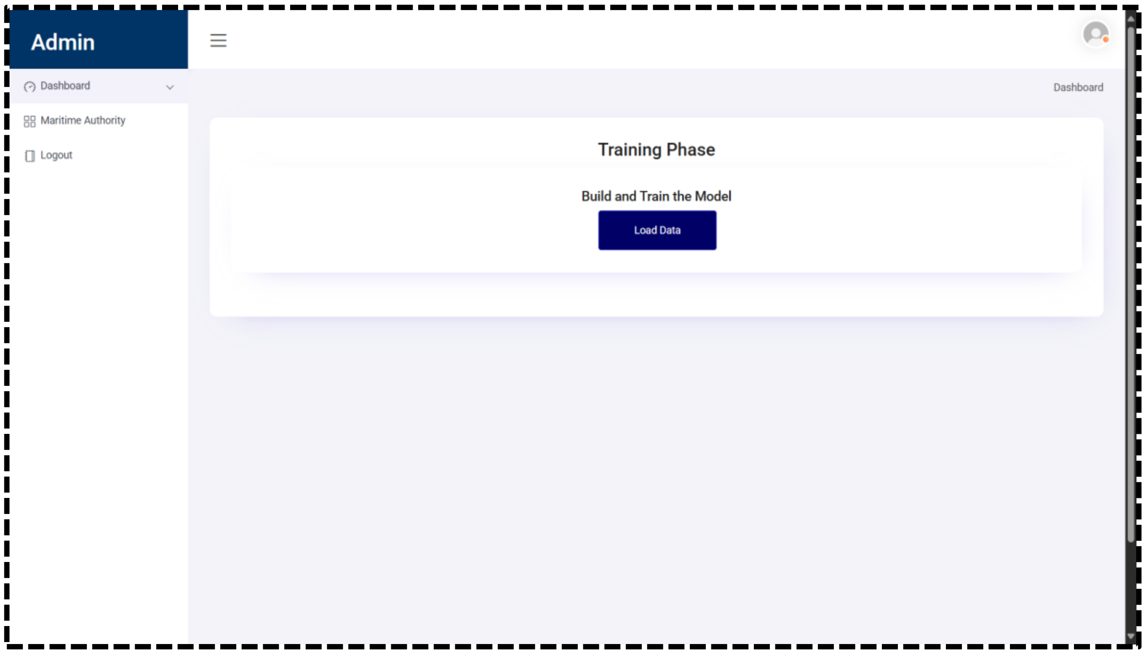
```
if ss[13]=="rain":
if float(ss[12])>0 and float(ss[11])>0:
r+=1
```

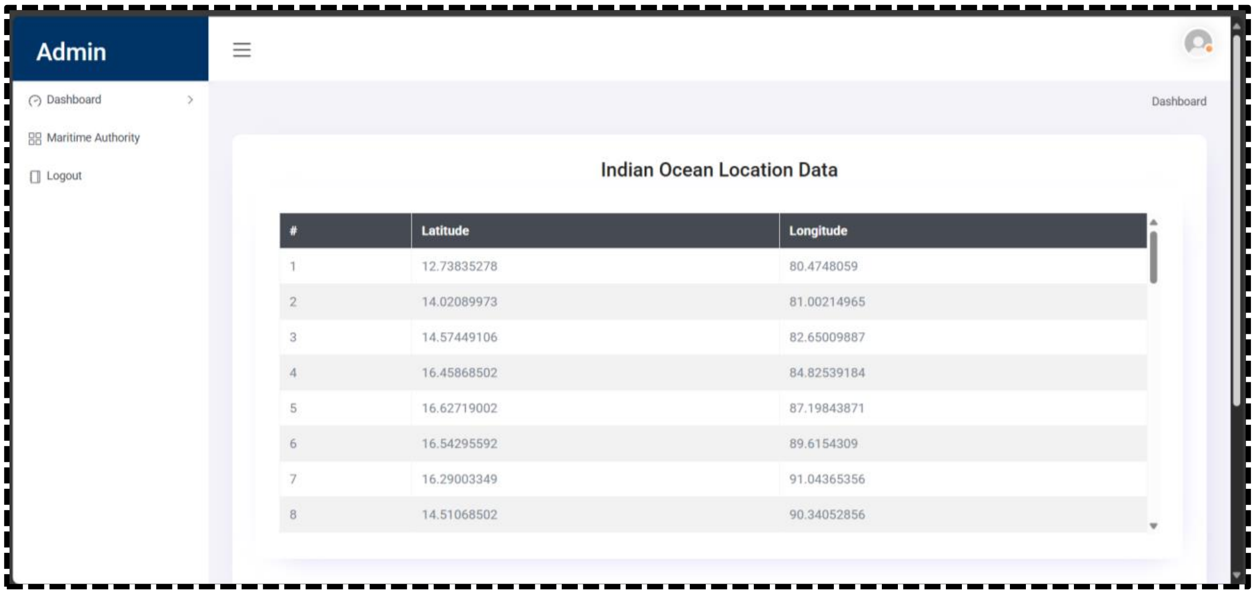
Fisherman Trip Request

```
def fm_trip():
msg=""
uname=""
data1=[]
act=request.args.get("act")
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM fm_fisher where uname=%s",(uname,))
rs = mycursor.fetchone()
fid=str(rs[0])
if request.method=='POST':
sdate=request.form['sdate']
sh=request.form['sh']
sm=request.form['sm']
sd=sdate.split("-")
sdate1=sd[2]+"-"+sd[1]+"-"+sd[0]
stime=sh+":"+sm
mycursor.execute("SELECT max(id)+1 FROM fm_trip")
maxid = mycursor.fetchone()[0]
if maxid is None:
maxid=1
sql = "INSERT INTO fm_trip(id,fisher_id,sdate,stime,edate,etime,request_st,trip_st) VALUES
(%s, %s, %s, %s, %s, %s, %s, %s)"
val = (maxid,uname,sdate1,stime,'','','0','0')
mycursor.execute(sql, val)
mydb.commit()
```

SCREENSHOT

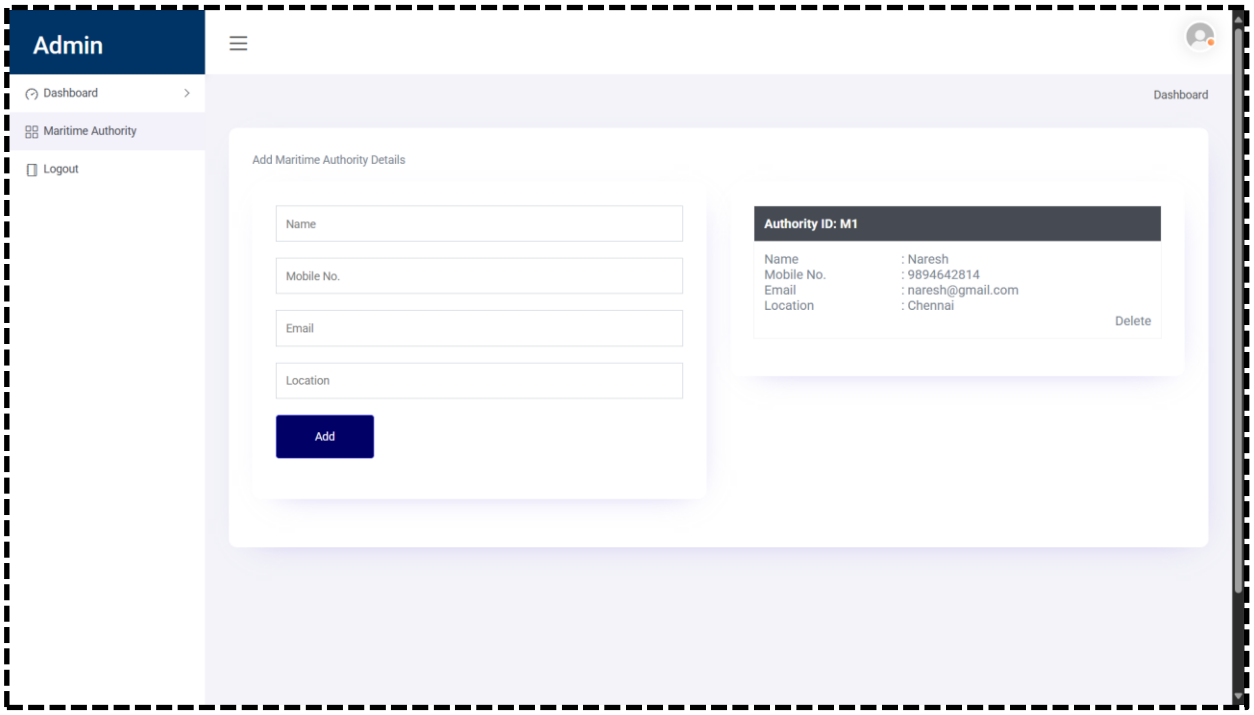

fisherman@info.com
+0011223495
Coastal Guard
Home
Fisherman
Maritime Authorities
Admin
Home
Admin
Admin Login
Username
admin
Password
Login


Admin
Dashboard
Maritime Authority
Logout
Dashboard
Training Phase
Build and Train the Model
Load Data



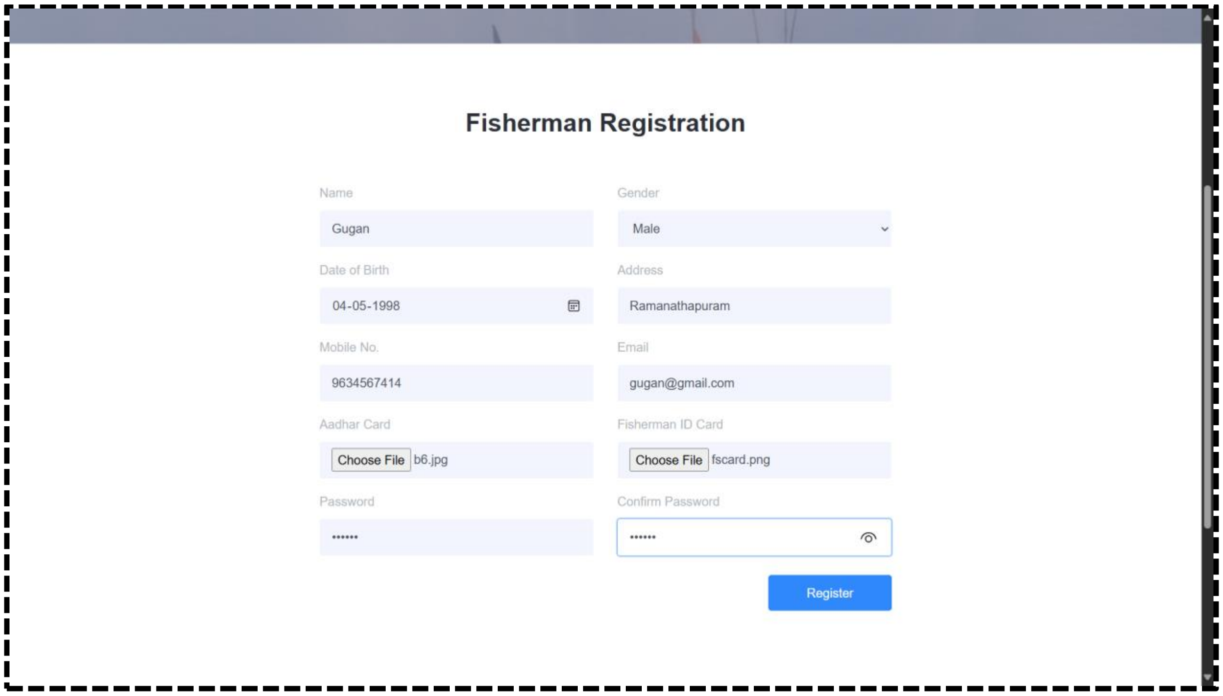
Indian Ocean Location Data

#	Latitude	Longitude
1	12.73835278	80.4748059
2	14.02089973	81.00214965
3	14.57449106	82.65009887
4	16.45868502	84.82539184
5	16.62719002	87.19843871
6	16.54295592	89.6154309
7	16.29003349	91.04365356
8	14.51068502	90.34052856

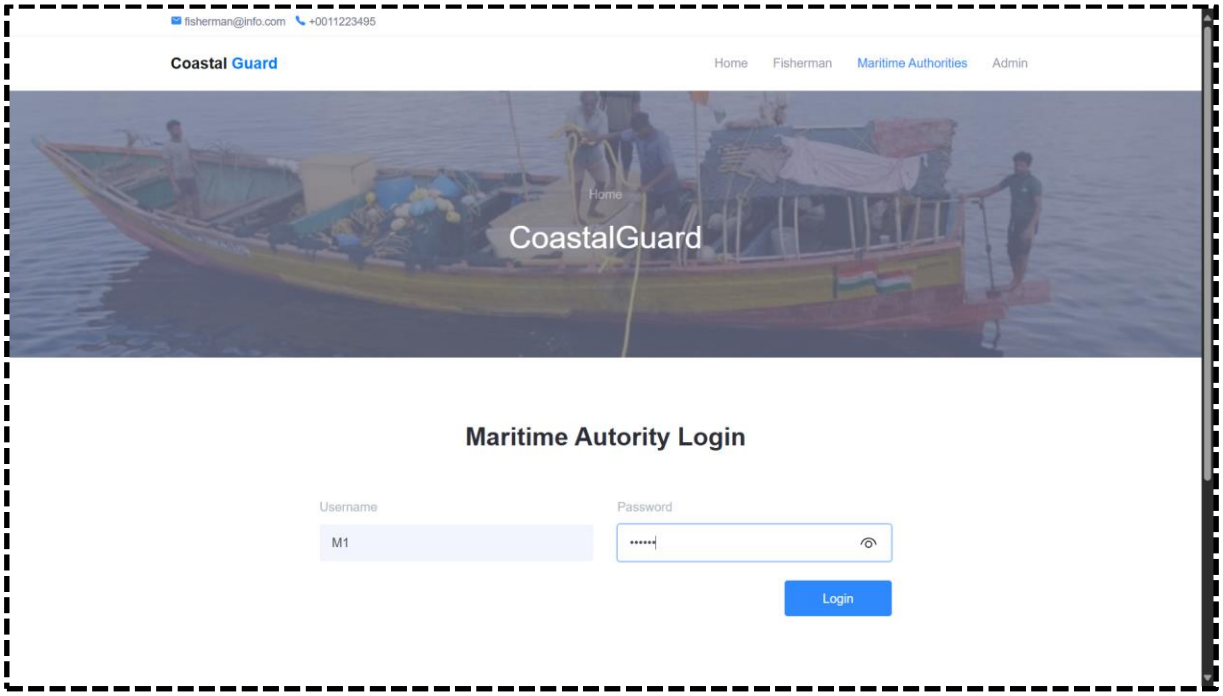


Authority ID: M1

Name	: Naresh
Mobile No.	: 9894642814
Email	: naresh@gmail.com
Location	: Chennai

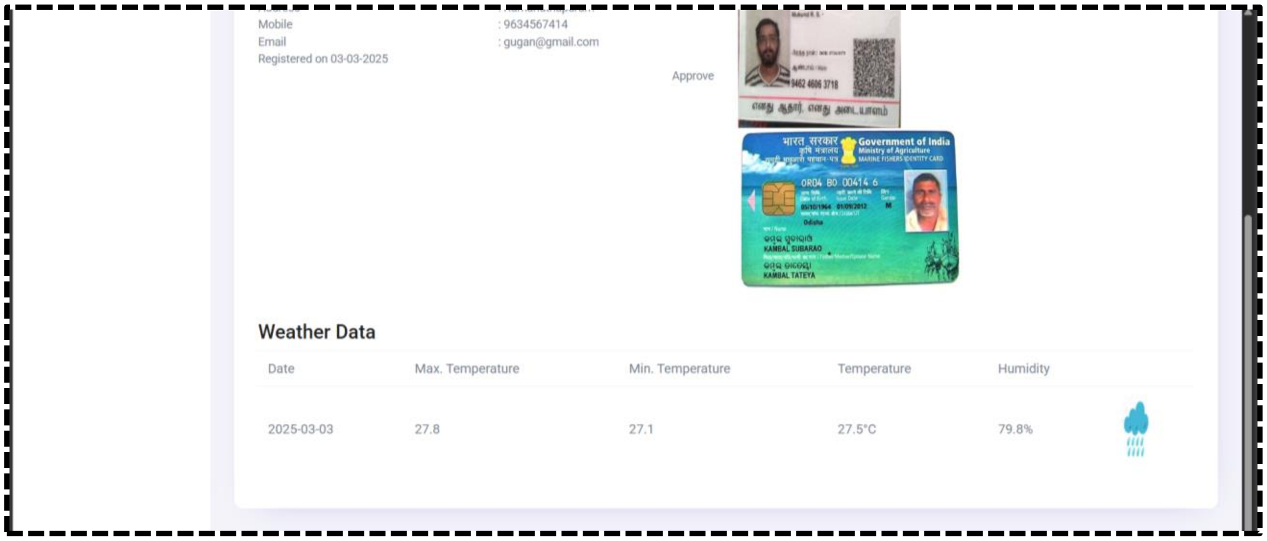

Fisherman Registration

Name	Gender
Gugan	Male
Date of Birth	Address
04-05-1998	Ramanathapuram
Mobile No.	Email
9634567414	gugan@gmail.com
Aadhar Card	Fisherman ID Card
Choose File b6.jpg	Choose File fscard.png
Password	Confirm Password
••••••	••••••


fisherman@info.com +0011223495

Maritime Autority Login

Username	Password
M1	••••••

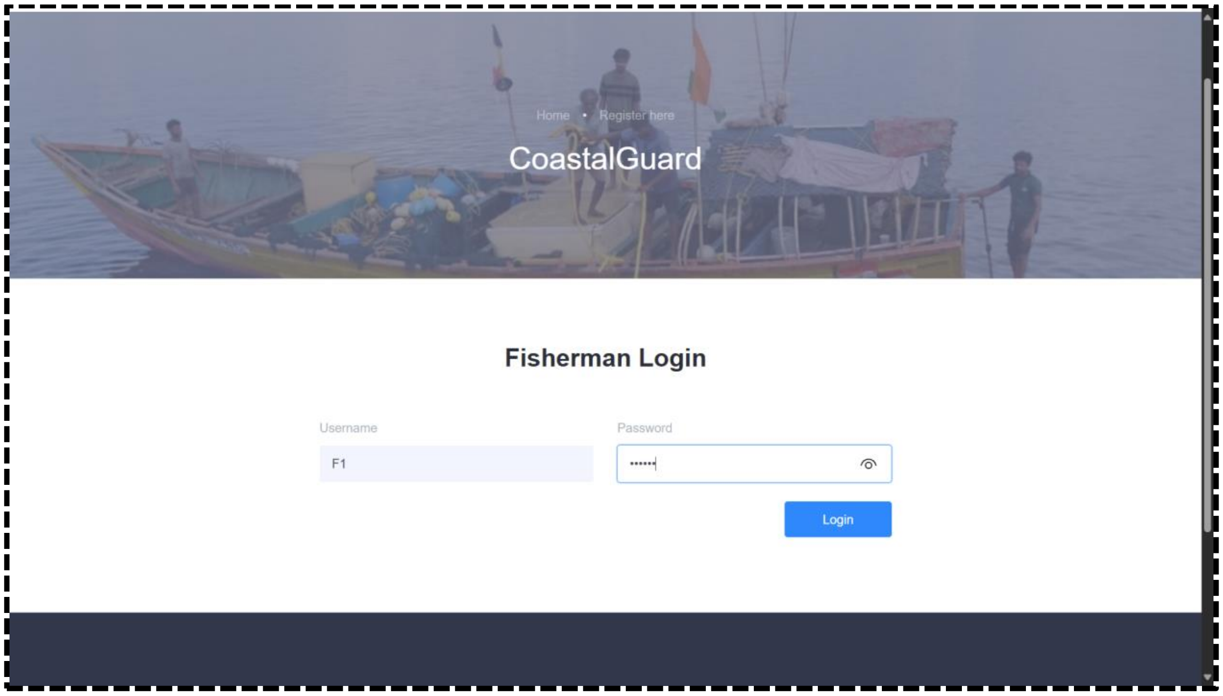


Mobile : 9634567414
Email : gugan@gmail.com
Registered on 03-03-2025

Approve

Weather Data

Date	Max. Temperature	Min. Temperature	Temperature	Humidity
2025-03-03	27.8	27.1	27.5°C	79.8%

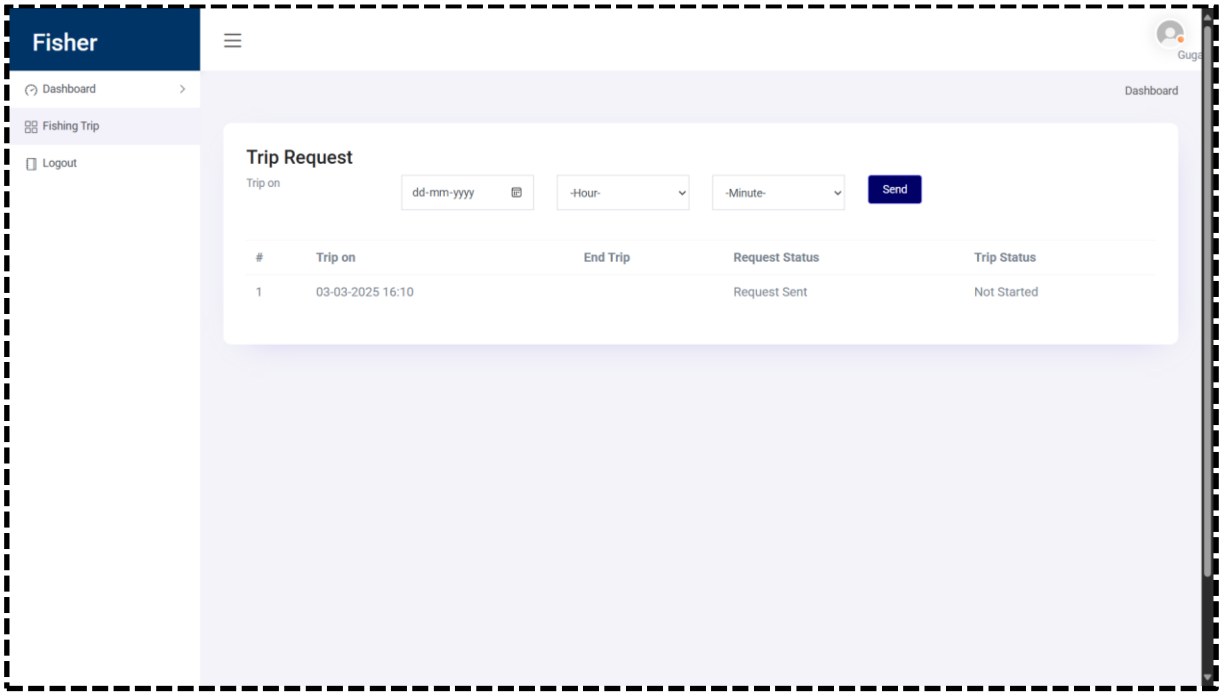


CoastalGuard

Fisherman Login

Username: F1

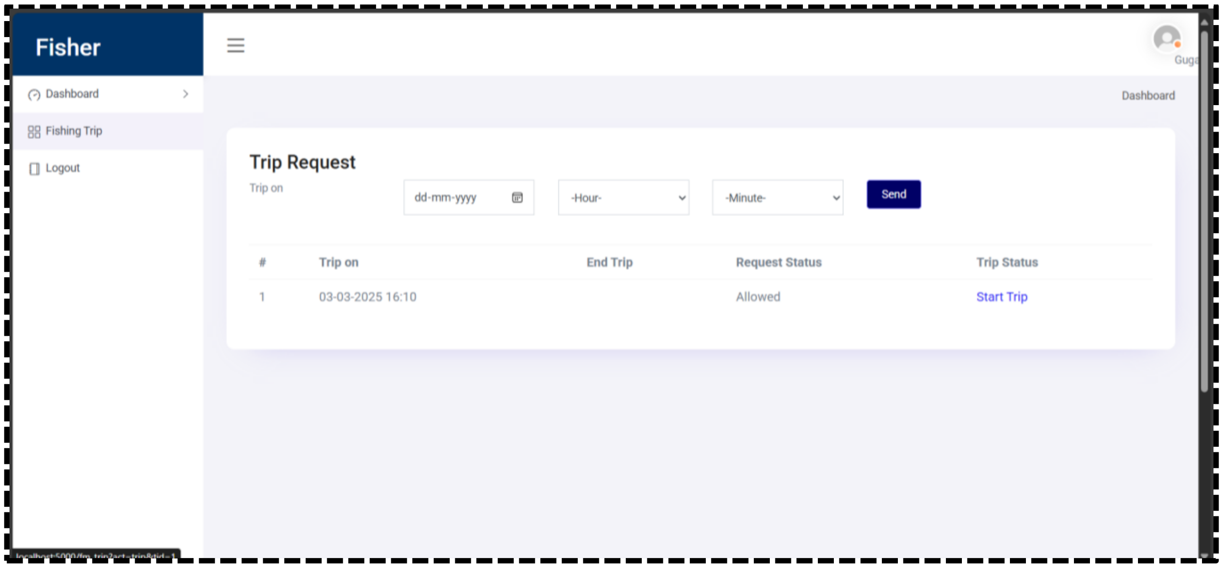
Password: ••••••



Trip Request

Trip on dd-mm-yyyy -Hour- -Minute-

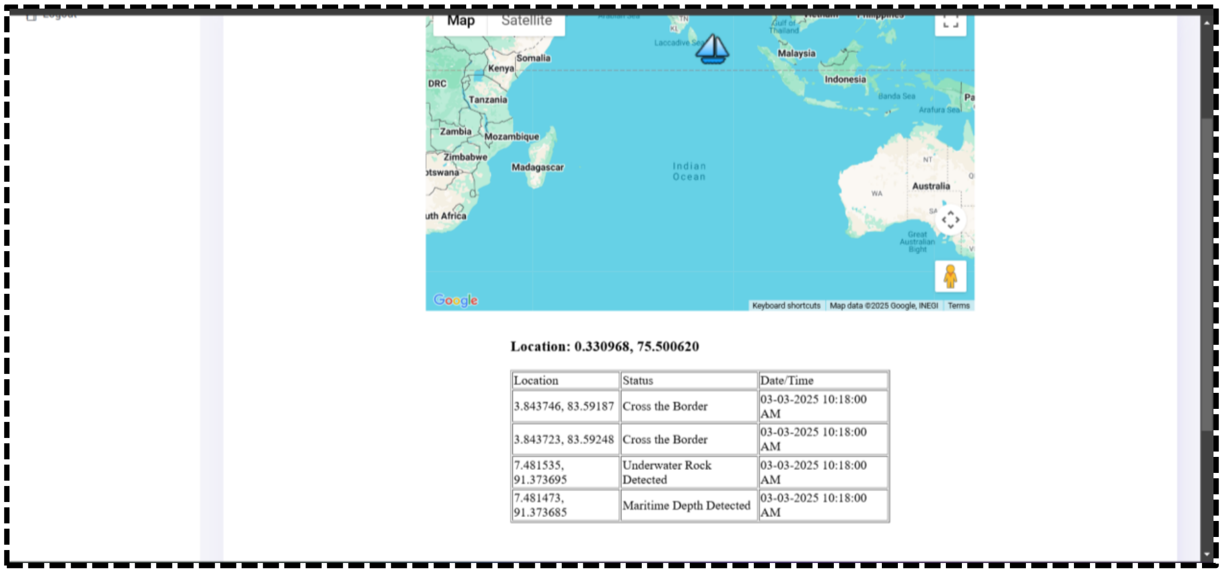
#	Trip on	End Trip	Request Status	Trip Status
1	03-03-2025 16:10		Request Sent	Not Started



Trip Request

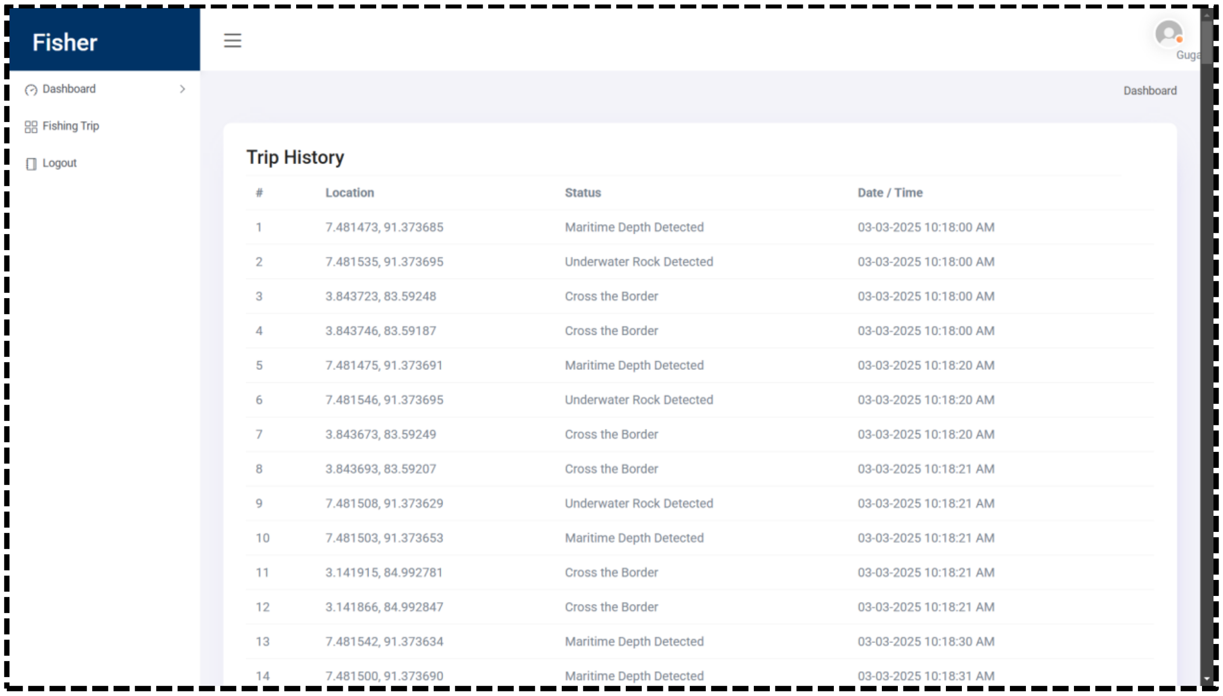
Trip on dd-mm-yyyy -Hour- -Minute-

#	Trip on	End Trip	Request Status	Trip Status
1	03-03-2025 16:10		Allowed	Start Trip



Location: 0.330968, 75.500620

Location	Status	Date/Time
3.843746, 83.59187	Cross the Border	03-03-2025 10:18:00 AM
3.843723, 83.59248	Cross the Border	03-03-2025 10:18:00 AM
7.481535, 91.373695	Underwater Rock Detected	03-03-2025 10:18:00 AM
7.481473, 91.373685	Maritime Depth Detected	03-03-2025 10:18:00 AM



Trip History

#	Location	Status	Date / Time
1	7.481473, 91.373685	Maritime Depth Detected	03-03-2025 10:18:00 AM
2	7.481535, 91.373695	Underwater Rock Detected	03-03-2025 10:18:00 AM
3	3.843723, 83.59248	Cross the Border	03-03-2025 10:18:00 AM
4	3.843746, 83.59187	Cross the Border	03-03-2025 10:18:00 AM
5	7.481475, 91.373691	Maritime Depth Detected	03-03-2025 10:18:20 AM
6	7.481546, 91.373695	Underwater Rock Detected	03-03-2025 10:18:20 AM
7	3.843673, 83.59249	Cross the Border	03-03-2025 10:18:20 AM
8	3.843693, 83.59207	Cross the Border	03-03-2025 10:18:21 AM
9	7.481508, 91.373629	Underwater Rock Detected	03-03-2025 10:18:21 AM
10	7.481503, 91.373653	Maritime Depth Detected	03-03-2025 10:18:21 AM
11	3.141915, 84.992781	Cross the Border	03-03-2025 10:18:21 AM
12	3.141866, 84.992847	Cross the Border	03-03-2025 10:18:21 AM
13	7.481542, 91.373634	Maritime Depth Detected	03-03-2025 10:18:30 AM
14	7.481500, 91.373690	Maritime Depth Detected	03-03-2025 10:18:31 AM

REFERENCES

JOURNAL REFERENCES

1. Muhammad Kahar and Mani Broto, "Management for Alleviating Poverty Among Fishermen Through Empowering Female Fishermen in North Kalimantan", *KnE Social Sciences.*, 2023.
2. BA. Anandh, R Sakthivel et al., "Alert system of fisherman for secure navigation", *Journal of Physics: Conference Series*, 2022.
3. R. Tharwin Kumar, *Article Title: Fisherman Border Alert System Based on IOT Fisherman Border Alert System Based on IOT*, vol. 1, pp. 37-44, 2022.
4. Tk Kesavan and K Lakshmi, *GSM-Based Smart Marine Tracking System for Location Monitoring and Emergency Protection*, 2022.
5. Durka Devi Jayaram, Saravana Kumar Jayaprabha and John Clement Sunder, "FISHERMEN BORDER SECURITY ALERT SYSTEM USING IOT", *International Research Journal of Modernization in Engineering Technology and Science*, 2021.
6. S. V. S. N. Murthy, B. V. V. Satyanarayana and C. V. V. S. Srinivas, "Location Tracking and Warning System of a Ship using Arduino", *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1786-1790, 2021.
7. T. Jayaram, D. Durkka Devi and S. Jayaprabha, "Fishermen Border Security Alert System Using IOT", *International Research Journal of Modernization in Engineering Technology and Science*, vol. 3, no. 04, Apr. 2021.

BOOK REFERENCES

1. "Python Crash Course" by Eric Matthes (Python Crash Course)
2. "Learning MySQL: Get a Handle on Your Data" by Russell J.T. Dyer (Learning MySQL)
3. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron (Hands-On Machine Learning)
4. "Python for Data Analysis" by Wes McKinney (Python for Data Analysis)
5. "WampServer 2: Manually Installing the Apache, MySQL, and PHP" by Dr. James R. Small (WampServer 2)
6. "Bootstrap 4 Quick Start: Responsive Web Design and Development with Bootstrap 4" by Jacob Lett (Bootstrap 4 Quick Start)

7. "Fluent Python: Clear, Concise, and Effective Programming" by Luciano Ramalho (Fluent Python)

WEB REFERENCES

1. Flask Documentation: Official documentation for Flask web framework - https://flask.palletsprojects.com/en/2.0.x/
2. MySQL Documentation: Official documentation for MySQL database management system - https://dev.mysql.com/doc/
3. TensorFlow Documentation: Official documentation for TensorFlow deep learning framework - https://www.tensorflow.org/guide
4. Pandas Documentation: Official documentation for Pandas data analysis library - https://pandas.pydata.org/docs/
5. Scikit-learn Documentation: Official documentation for Scikit-learn machine learning library - https://scikit-learn.org/stable/documentation.html
6. Matplotlib Documentation: Official documentation for Matplotlib data visualization library - https://matplotlib.org/stable/contents.html
7. NumPy Documentation: Official documentation for NumPy numerical computing library - https://numpy.org/doc/stable/