A

Major Project

On

PREDICTING URBAN WATER QUALITY WITH UBIQUITOUS DATA – A DATA DRIVEN APPROACH

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

KHAJA MUBASHIRUDDIN (207R1A05L6)
P. BHARATH SIMHA REDDY (207R1A05N5)
GADDE SAYI KHUSHHAL (207R1A05L0)

Under the Guidance of

S. APARNA

(Assistant Professor)

GROUP OF INSTITUTIONS

EXPLORE TO INVENT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V) , ,Medchal Road, Hyderabad-501401.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR
GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CERTIFICATE

This is to certify that the project entitled **“PREDICTING URBAN WATER QUALITY WITH UBIQUITOUS DATA – A DATA DRIVEN APPROACH”** being submitted by **KHAJA MUBASHIRUDDIN (207R1A05L6), P. BHARATH SIMHA REDDY (207R1A05N5) & GADDE SAYI KHUSHHAL (207R1A05L0)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-2024.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

S. Aparna
(Assistant Professor)
INTERNAL GUIDE

DR. A. RAJIREDDY
DIRECTOR

DR. K. SRUJAN RAJU
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on______________________

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **S. Aparna,** Associate Professor for her exemplary guidance, monitoring, and constant encouragement throughout the project work. The blessing, help, and guidance given by her shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **G.Vinesh Shanker, Dr. J. Narasimharao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

KHAJA MUBASHIRUDDIN (207R1A05L6)
P. BHARATH SIMHA REDDY (207R1A05N5)
GADDE SAYI KHUSHHAL (207R1A05L0)

ABSTRACT

Urban water quality is of great importance to our daily lives. Prediction of urban water quality help control water pollution and protect human health. However, predicting the urban water quality is a challenging task since the water quality varies in urban spaces non-linearly and depends on multiple factors, such as meteorology, water usage patterns, and land uses. In this work, we forecast the water quality of a station over the next few hours from a data-driven perspective, using the water quality data and water hydraulic data reported by existing monitor stations and a variety of data sources we observed in the city, such as meteorology, pipe networks, structure of road networks, and point of interests (POIs).

First, we identify the influential factors that affect the urban water quality via extensive experiments. Second, we present a multi-task multi-view learning method to fuse those multiple datasets from different domains into a unified learning model. We evaluate our method with real-world datasets, and the extensive experiments verify the advantages of our method over other baselines and demonstrate the effectiveness of our approach. Machine learning is an important component of the growing field of data science.

Through the use of statistical methods, different type of algorithms is trained to make classifications or predictions and to uncover key insights in this project. These insights subsequently drive decision-making within applications and businesses, ideally impacting key growth metrics. Machine learning algorithms build a model based on this project data, known as training data, to make predictions or decisions without being explicitly programmed to do so.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

Urban water is a vital resource that affects various aspects of human, health and urban lives. People living in major cities are increasingly concerned about the urban water quality, calling for technology that can monitor and predict the water quality in real-time throughout the city. Urban water quality, which serves as "a powerful environmental determinant" and "a foundation for the prevention and control of waterborne diseases", refers to the physical, chemical and biological characteristics of a water body, and several chemical indexes (such as residual chlorine, turbidity and pH) can be used as effective measurements for the water quality in current urban water distribution systems.

1.2 PROJECT PURPOSE

The purpose of this project is to develop a data-driven approach for predicting urban water quality. Recognizing the significance of urban water quality in daily life and its impact on public health, the project aims to overcome the challenges posed by non-linear variations in water quality within urban spaces. By leveraging data from diverse sources, including monitor stations, meteorology, pipe networks, and points of interest, the project seeks to identify influential factors and create a unified learning model. The ultimate goal is to forecast water quality at specific stations over the next few hours, contributing to effective water pollution control and the protection of human health in urban environments.

1.3 PROJECT FEATURES

The project features include the identification of influential factors impacting urban water quality through extensive experiments. It employs a multi-task multi-view learning method to integrate data from various domains, such as water quality, hydraulic data, meteorology, pipe networks, road structures, and points of interest. The project focuses on a data-driven perspective, emphasizing the practical application of machine learning to predict urban water quality accurately.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “What must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

This project addresses the complex problem of predicting urban water quality. The challenge lies in the non-linear variations influenced by factors like meteorology and land use. The goal is to forecast water quality at specific stations by employing a data-driven approach, integrating data from various sources such as monitor stations, meteorology, pipe networks, and points of interest. The project aims to identify influential factors, employ multi-task multi-view learning, and validate its effectiveness through real-world dataset experiments.

2.2 EXISTING SYSTEM

Several studies in the environmental science have been tried to analyze the water quality problems via data-driven based approaches, and those studies covers a range of topics, from the physical process analysis in the river basin, to the analysis of concurrent input and output time series. The approaches adopted in these studies include instance-based learning models (e.g., kNN) as well as neural network models (e.g., ANN). The traditional baselines we see are ARMA, Kalman filter, and ANN Algorithms, leading to a maximum accuracy of 60%. These traditional baselines use static data instead of real-time data as well as use only single algorithm approaches.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

- The system supports only static data.
- The accuracy of the existing systems is around 60%

2.3 PROPOSED SYSTEM

In the proposed system for Predicting Urban Water Quality with Ubiquitous Data, we've developed a new way to predict water quality at different locations using data from various sources. This method isn't just limited to cities but can also be applied to similar prediction problems in other urban areas. We've pinpointed factors that affect water quality, like nearby places, pipe networks, time of day, and weather conditions, which can improve predictions not just for us but for others working on water quality issues. Our method outperforms traditional techniques like ARMA, Kalman filter, and artificial neural networks (ANN), as shown by our tests using real data from India. By using simple methods like Logistic Regression, Support Vector Machines (SVM), and KNN, we can make accurate predictions and uncover insights to benefit urban life and society.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- We collect water quality data from many water quality monitoring stations in India. It comprises residual chlorine (RC), turbidity (TU), and pH.
- In this project, we only use RC as the index for water quality, since RC is the most important and effective measurement for water quality in the current urban water distribution system.
- Leading to a much-improved accuracy in predicting the water quality, approximately 78%.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal isput forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited**.** The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 SOCIAL FEASIBILITY

The focal point of our study revolves around evaluating the degree of acceptance exhibited by users towards the system. This encompasses not only the process of familiarizing users with the system but also ensuring their proficiency in its operation. It is imperative that users do not perceive the system as a threat; rather, they should regard it as an indispensable tool that enhances their workflow. The level of acceptance among users is intricately tied to the methodologies employed to educate them about the system and facilitate their acquaintance with its functionalities. Our objective is to cultivate a sense of confidence among users, empowering them to provide constructive feedback that enriches the system's utility. After all, users are the ultimate beneficiaries of the system, and their input is invaluable in refining its effectiveness.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements:

- **System** : Pentium IV
- **Hard Disk** : 20 GB.
- **RAM** : 4GB(min)

2.5.2 SOFTWARE REQUIREMENTS:

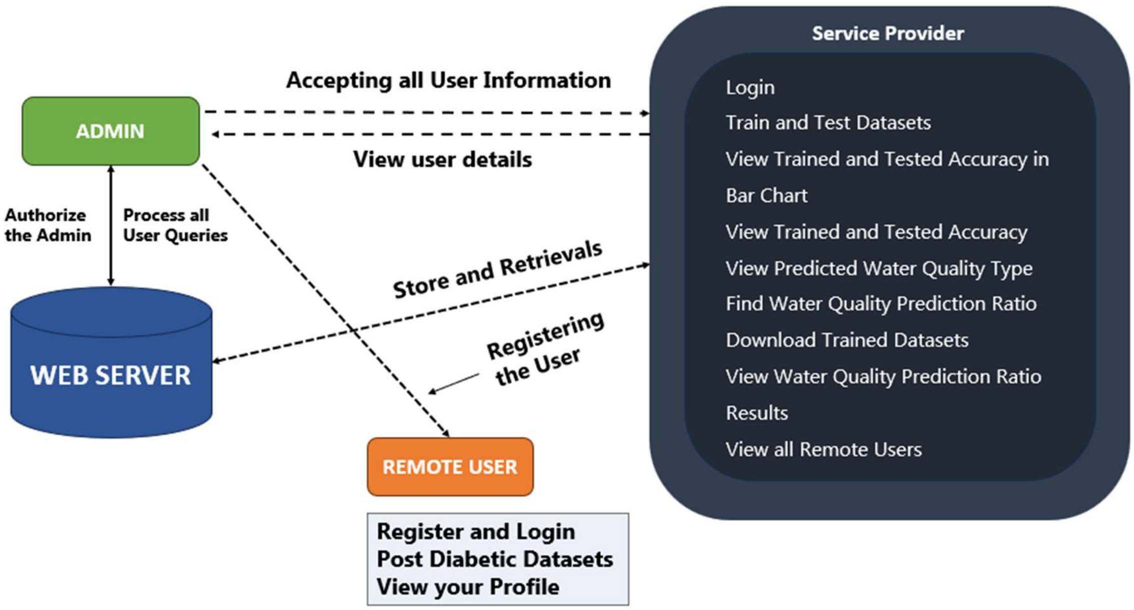
Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements:

- **Coding Language** : Python3.7
- **Database** : MySQL (XAMPP Server)
- **Front End** : HTML, CSS, JavaScript
- **Back End** : Django-ORM

3.ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE



This project architecture shows the procedure followed for classification, starting from input to final prediction.

Figure 3.1: Project Architecture of the Proposed Schema for Predicting Urban Water Quality with Ubiquitous Data – A Data Driven Approach

3.2 DESCRIPTION

The system architecture consists of four main components: Admin, Web Server, Remote User, and Service Provider, enabling water quality prediction and management. Admin oversees system management, while the Web Server acts as a communication hub. Remote Users register, login, input data, and manage profiles, while the Service Provider handles dataset management, model training, testing, and prediction. Users interact with the Service Provider for various tasks, including data upload, viewing accuracy results, and predicting water quality types. Admins monitor and manage Remote Users.

3.3 USE CASE DIAGRAM:

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

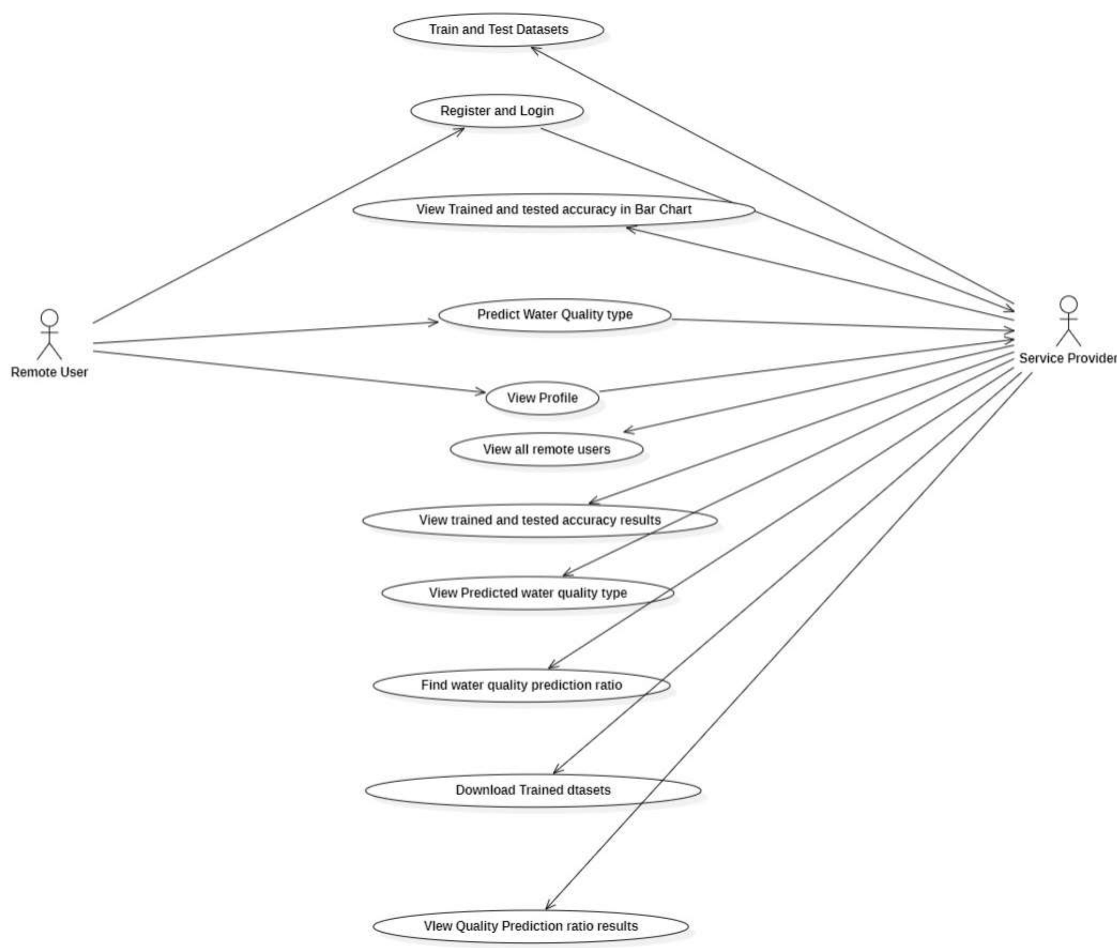
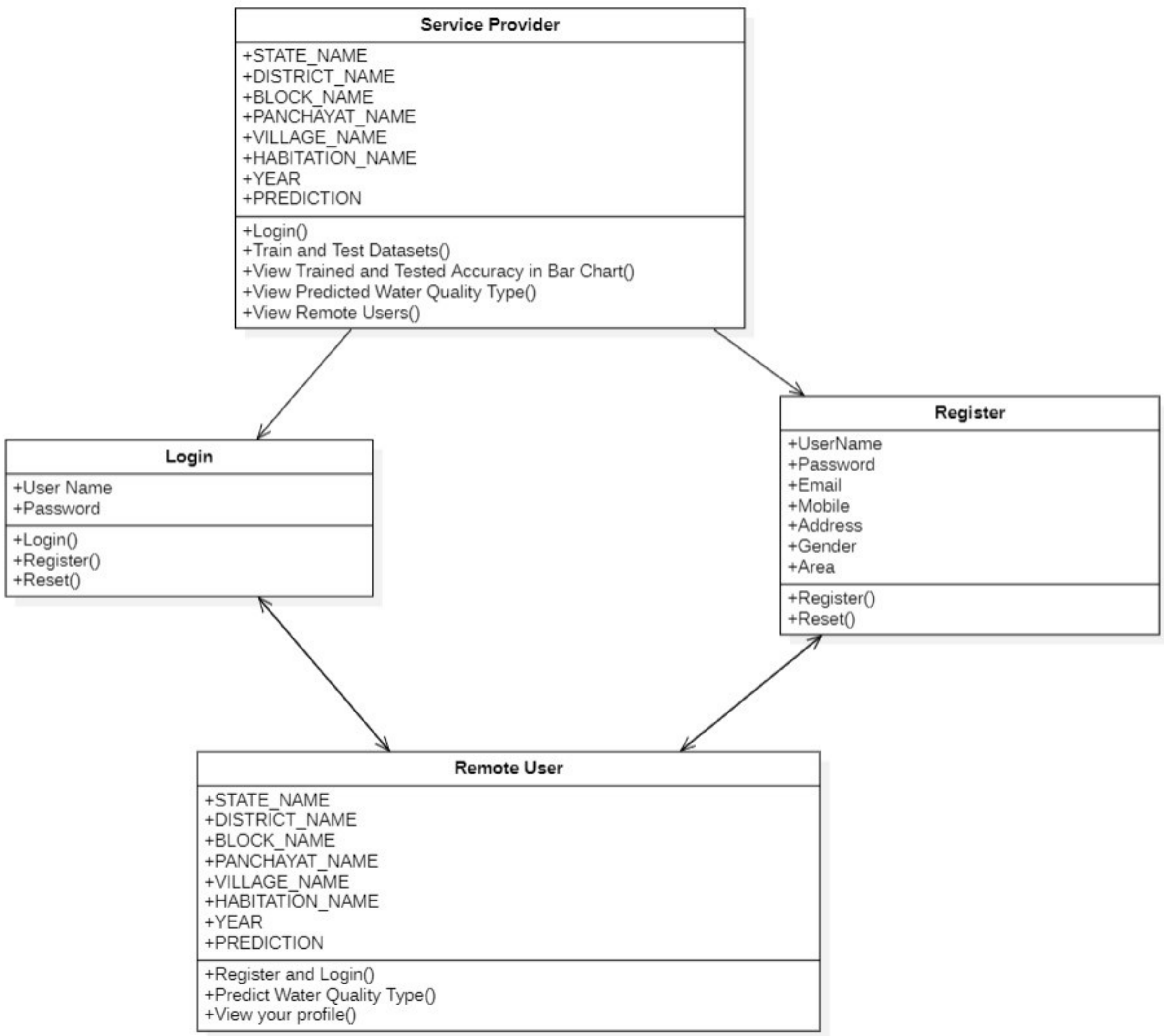


Figure 3.2: Use Case Diagram for Predicting Urban Water Quality with Ubiquitous Data – A Data Driven Approach

3.4 CLASS DIAGRAM



Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Figure 3.3: Class Diagram for Predicting Urban Water Quality with Ubiquitous Data – A Data Driven Approach

3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

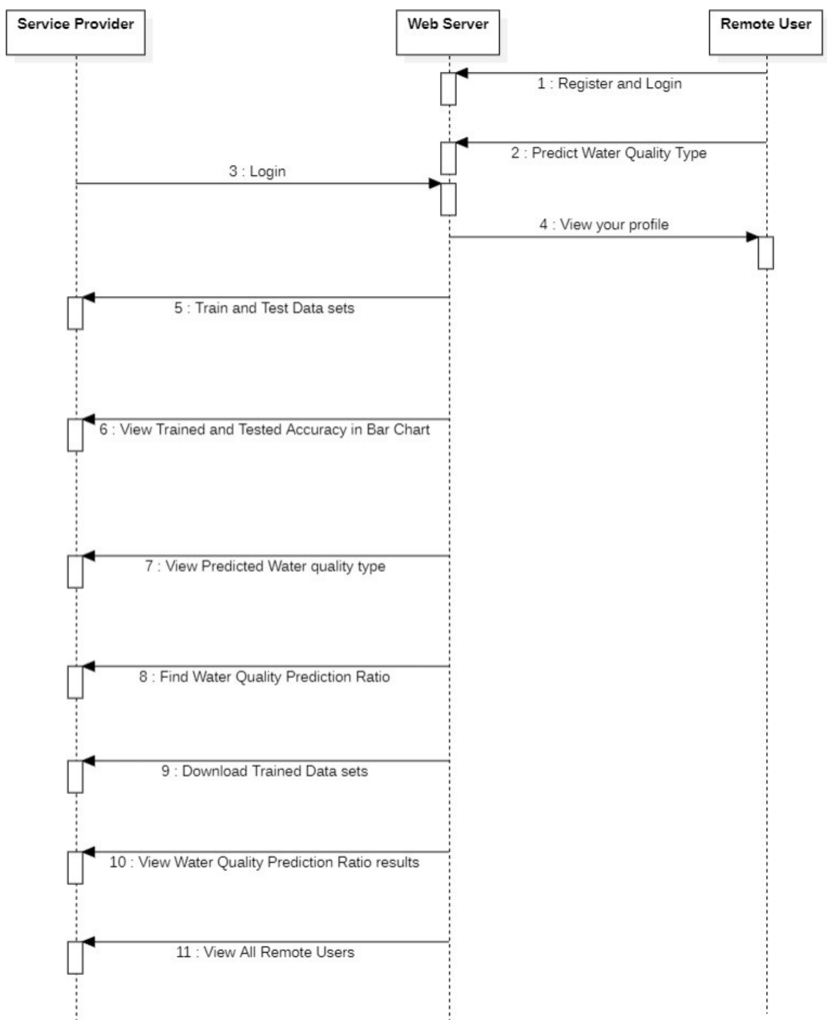


Figure 3.4: Sequence Diagram for Predicting Urban Water Quality with Ubiquitous Data – A Data Driven Approach

3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

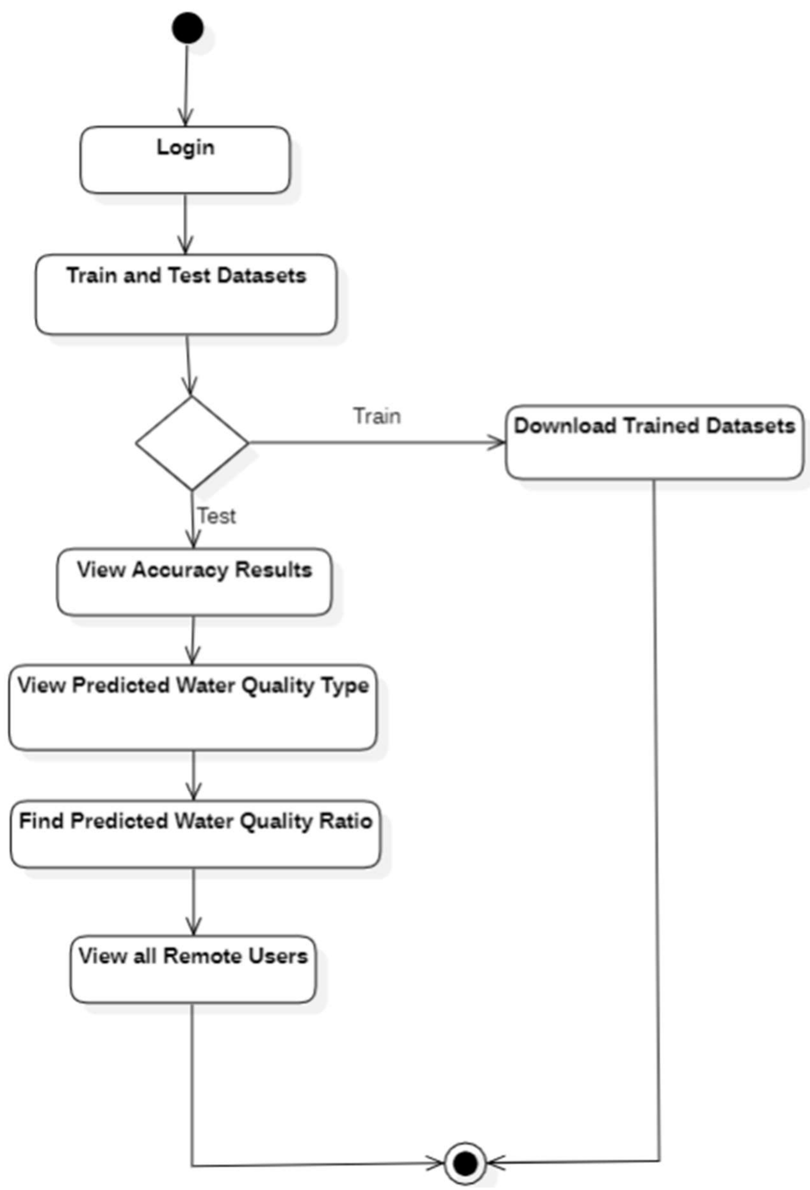


Figure 3.5: Activity Diagram for Predicting Urban Water Quality with Ubiquitous Data – A Data Driven Approach

4. IMPLEMENTATION

4.1 SAMPLE CODE

```
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime
import xlwt
from django.http import HttpResponse

import pandas as pd
import warnings
warnings.filterwarnings("ignore")

from wordcloud import WordCloud,STOPWORDS
stopwords = set(STOPWORDS)

import re
from collections import Counter

from sklearn.metrics import classification_report,accuracy_score,confusion_matrix



# Create your views here.
from Remote_User.models import
ClientRegister_Model,water_quality_type,detection_ratio,detection_accuracy


def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "Admin" and password =="Admin":
            detection_accuracy.objects.all().delete()
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')
```

```
def Find_Water_Quality_Predicted_Ratio(request):
    detection_ratio.objects.all().delete()
    ratio = ""
    kword = 'Salinity'
    print(kword)
    obj = water_quality_type.objects.all().filter(Q(Prediction=kword))
    obj1 = water_quality_type.objects.all()
    count = obj.count();
    count1 = obj1.count();
    ratio = (count / count1) * 100
    if ratio != 0:
        detection_ratio.objects.create(names=kword, ratio=ratio)

    ratio1 = ""
    kword1 = 'Fluoride'
    print(kword1)
    obj1 = water_quality_type.objects.all().filter(Q(Prediction=kword1))
    obj11 = water_quality_type.objects.all()
    count1 = obj1.count();
    count11 = obj11.count();
    ratio1 = (count1 / count11) * 100
    if ratio1 != 0:
        detection_ratio.objects.create(names=kword1, ratio=ratio1)

    ratio12 = ""
    kword12 = 'Iron'
    print(kword12)
    obj12 = water_quality_type.objects.all().filter(Q(Prediction=kword12))
    obj112 = water_quality_type.objects.all()
    count12 = obj12.count();
    count112 = obj112.count();
    ratio12 = (count12 / count112) * 100
    if ratio12 != 0:
        detection_ratio.objects.create(names=kword12, ratio=ratio12)
```

```
ratio123 = ""
    kword123 = 'Arsenic-Fully Polluted'
    print(kword123)
    obj123 = water_quality_type.objects.all().filter(Q(Prediction=kword123))
    obj1123 = water_quality_type.objects.all()
    count123 = obj123.count();
    count1123 = obj1123.count();
    ratio123 = (count123 / count1123) * 100
    if ratio123 != 0:
        detection_ratio.objects.create(names=kword123, ratio=ratio123)
obj = detection_ratio.objects.all()
    return render(request, 'SProvider/Find_Water_Quality_Predicted_Ratio.html', {'objs':
obj})


def View_Remote_Users(request):
            obj=ClientRegister_Model.objects.all()
            return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def ViewTrendings(request):
            topic =
        water_quality_type.objects.values('topics').annotate(dcount=Count('topics')).order_by('-
        dcount')
            return  render(request,'SProvider/ViewTrendings.html',{'objects':topic})

def charts(request,chart_type):
            chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
            return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def charts1(request,chart_type):
            chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
            return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def View_Predicted_Water_Quality(request):
            obj =water_quality_type.objects.all()
            return render(request, 'SProvider/View_Predicted_Water_Quality.html', {'list_objects':
        obj})

def charts1(request,chart_type):
            chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
            return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def View_Predicted_Water_Quality(request):
            obj =water_quality_type.objects.all()
            return render(request, 'SProvider/View_Predicted_Water_Quality.html', {'list_objects':
        obj})
```

```
def likeschart(request,like_chart):
    charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})


def Download_Trained_DataSets(request):

    response = HttpResponse(content_type='application/ms-excel')
    # decide file name
    response['Content-Disposition'] = 'attachment; filename="PredictedData.xls"'
    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')
    # adding sheet
    ws = wb.add_sheet("sheet1")
    # Sheet header, first row
    row_num = 0
    font_style = xlwt.XFStyle()
    # headers are bold
    font_style.font.bold = True
    # writer = csv.writer(response)
    obj = water_quality_type.objects.all()
    data = obj  # dummy method to fetch data.

    for my_row in data:
        row_num = row_num + 1

        ws.write(row_num, 0, my_row.State_Name, font_style)
        ws.write(row_num, 1, my_row.District_Name, font_style)
        ws.write(row_num, 2, my_row.Block_Name, font_style)
        ws.write(row_num, 3, my_row.Panchayat_Name, font_style)
        ws.write(row_num, 4, my_row.Village_Name, font_style)
        ws.write(row_num, 5, my_row.Habitation_Name, font_style)
        ws.write(row_num, 6, my_row.Year, font_style)
        ws.write(row_num, 7, my_row.Prediction, font_style)

    wb.save(response)
    return response
```

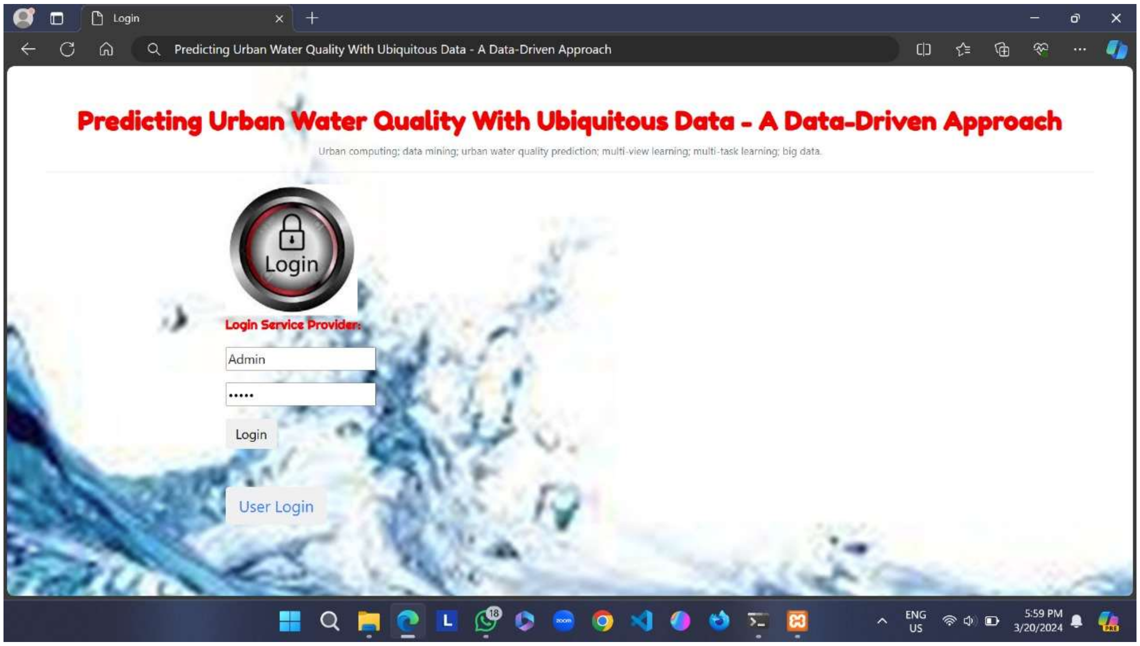
```
def train_model(request):
    detection_accuracy.objects.all().delete()
    df = pd.read_csv('Water_Quality_Datasets.csv',encoding='latin-1')

    def apply_results(results):
        if (results == 'Salinity'):
            return 0
        elif (results == 'Fluoride'):
            return 1
        elif (results == 'Iron'):
            return 2
        elif (results == 'Arsenic'):
            return 3
        elif (results == 'Nitrate'):
            return 4

    df['results'] = df['Quality_Parameter'].apply(apply_results)

    X = df['Habitation_Name']
    y = df['results']


    from sklearn.feature_extraction.text import CountVectorizer
    cv = CountVectorizer(lowercase=False, strip_accents='unicode', ngram_range=(1, 1))

    x = cv.fit_transform(X)


    models = []
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
    X_train.shape, X_test.shape, y_train.shape

    print("Naive Bayes")

    from sklearn.naive_bayes import MultinomialNB
```

```
NB = MultinomialNB()
    NB.fit(X_train, y_train)
    predict_nb = NB.predict(X_test)
    naivebayes = accuracy_score(y_test, predict_nb) * 100
    print("ACCURACY")
    print(naivebayes)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, predict_nb))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, predict_nb))
    detection_accuracy.objects.create(names="Naive Bayes", ratio=naivebayes)
    models.append(('naive_bayes', NB))

    # SVM Model
    print("SVM")
    from sklearn import svm

    lin_clf = svm.LinearSVC()
    lin_clf.fit(X_train, y_train)
    predict_svm = lin_clf.predict(X_test)
    svm_acc = accuracy_score(y_test, predict_svm) * 100
    print("ACCURACY")
    print(svm_acc)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, predict_svm))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, predict_svm))
    detection_accuracy.objects.create(names="SVM", ratio=svm_acc)
    models.append(('SVM', lin_clf))

    print("Logistic Regression")
```

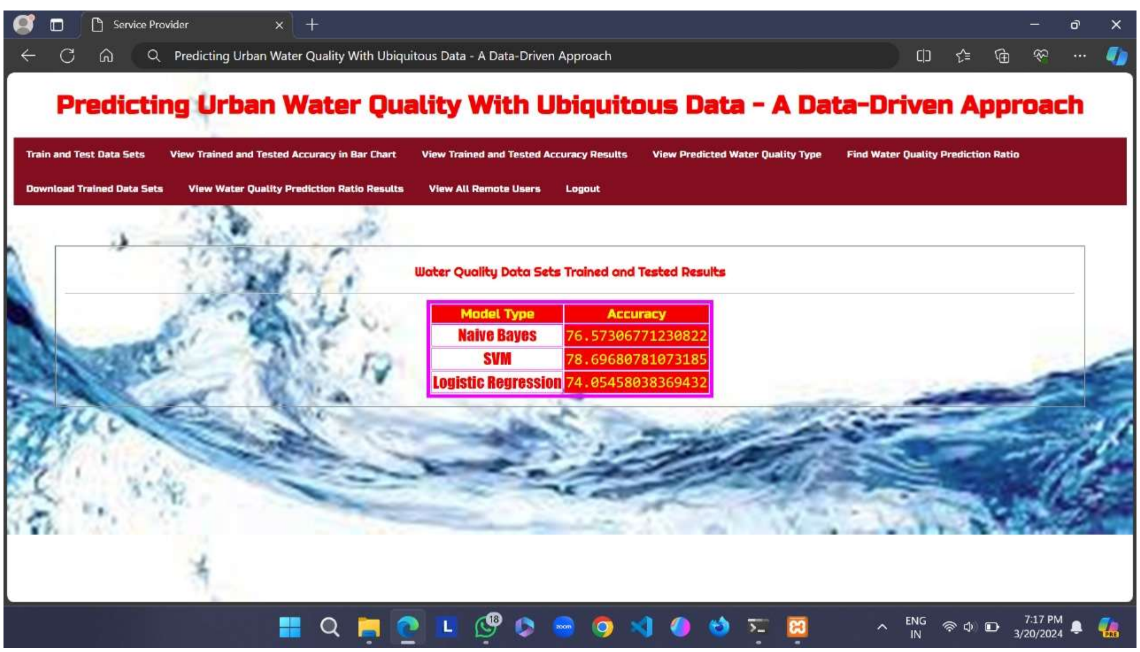
```
from sklearn.linear_model import LogisticRegression

    reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
    y_pred = reg.predict(X_test)
    print("ACCURACY")
    print(accuracy_score(y_test, y_pred) * 100)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, y_pred))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, y_pred))
    detection_accuracy.objects.create(names="Logistic Regression",
ratio=accuracy_score(y_test, y_pred) * 100)
    models.append(('LogisticRegression', reg))

    csv_format = 'Results.csv'
    df.to_csv(csv_format, index=False)
    df.to_markdown

    obj = detection_accuracy.objects.all()
    return render(request,'SProvider/train_model.html', {'objs': obj})
```

5. SCREENSHOTS

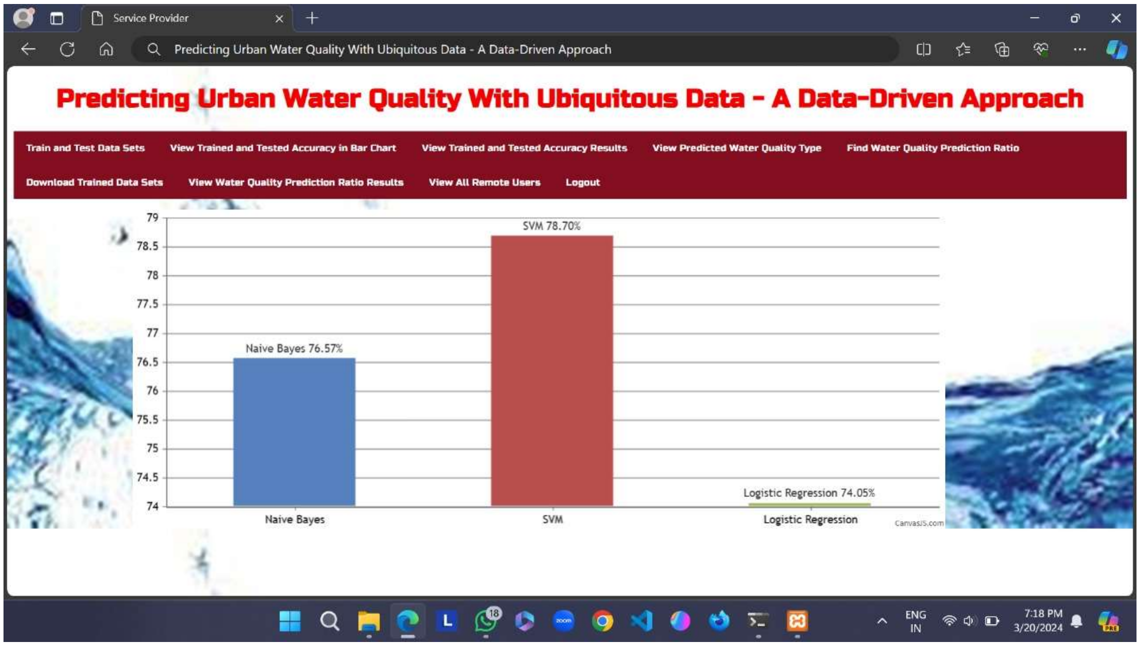
Screenshot 5.1: Service Provider Login



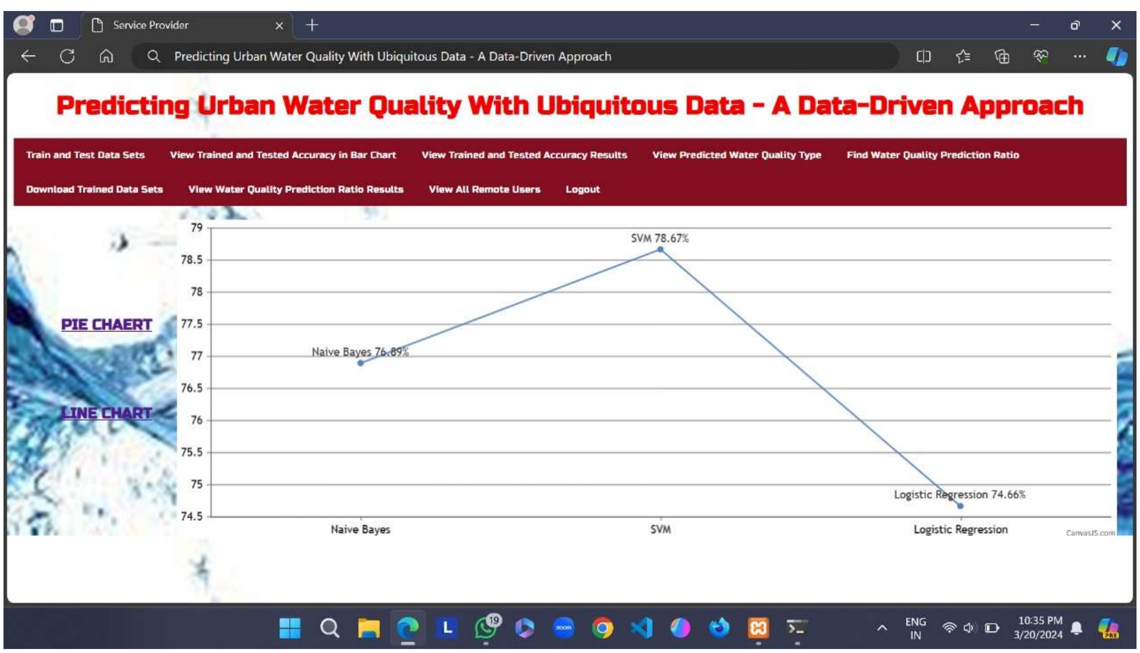
Screenshot 5.2: Train and Test Datasets



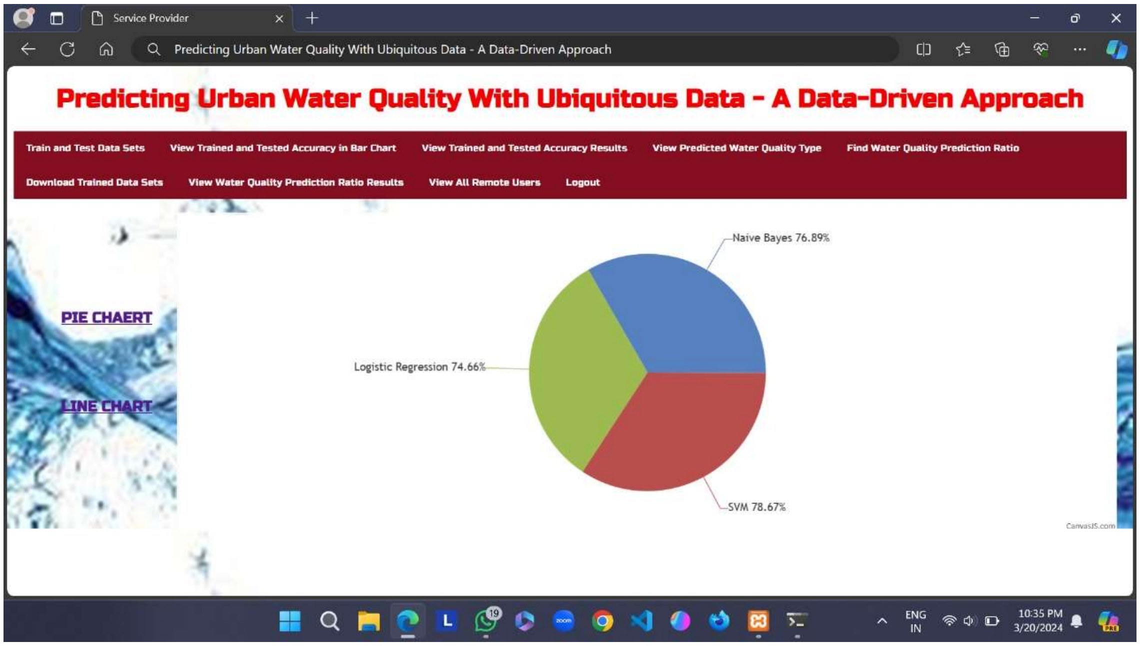
Screenshot 5.3: View Trained and Tested Accuracy in Bar Chart



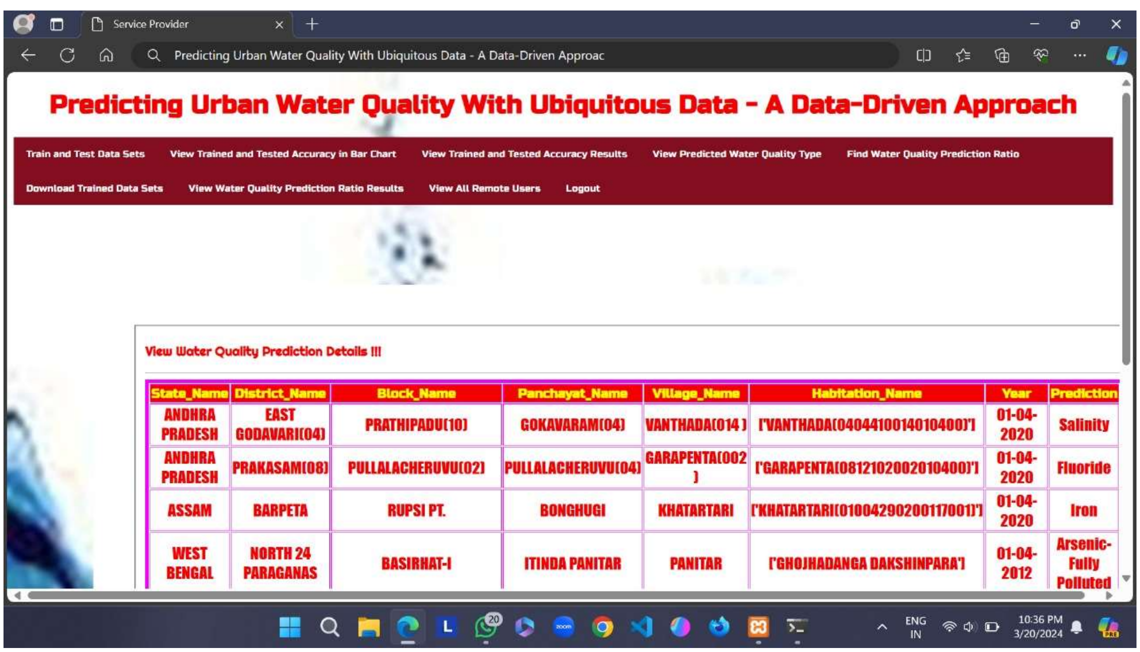
Screenshot 5.4: View Trained and Tested Accuracy in Line Chart



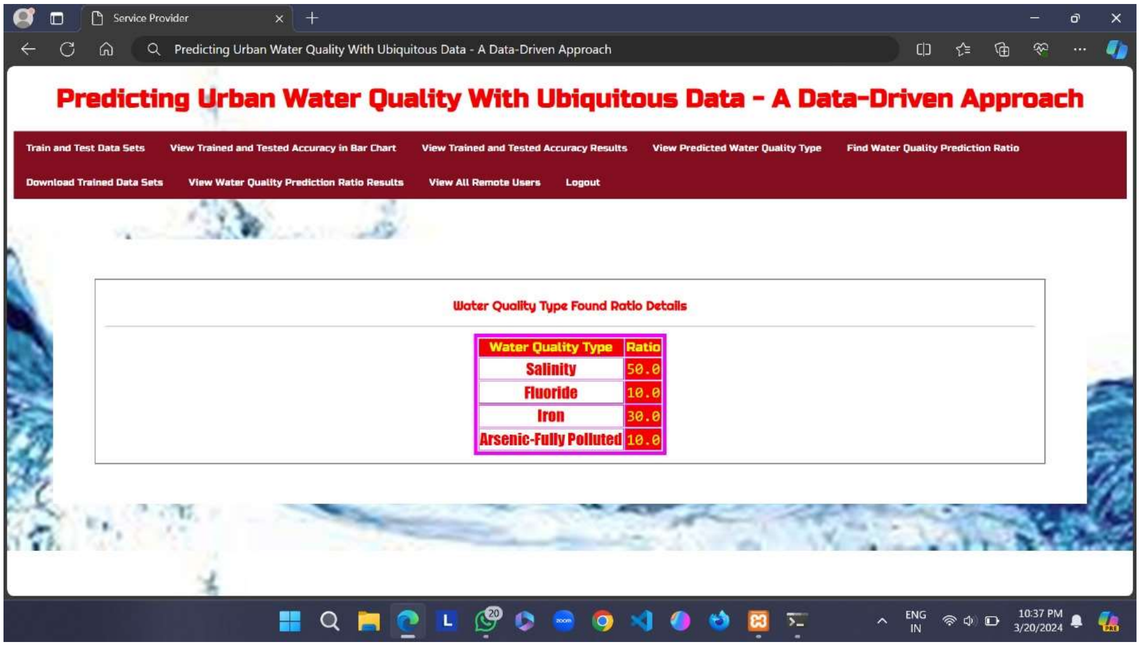
Screenshot 5.5: View Trained and Tested Accuracy in Pie Chart



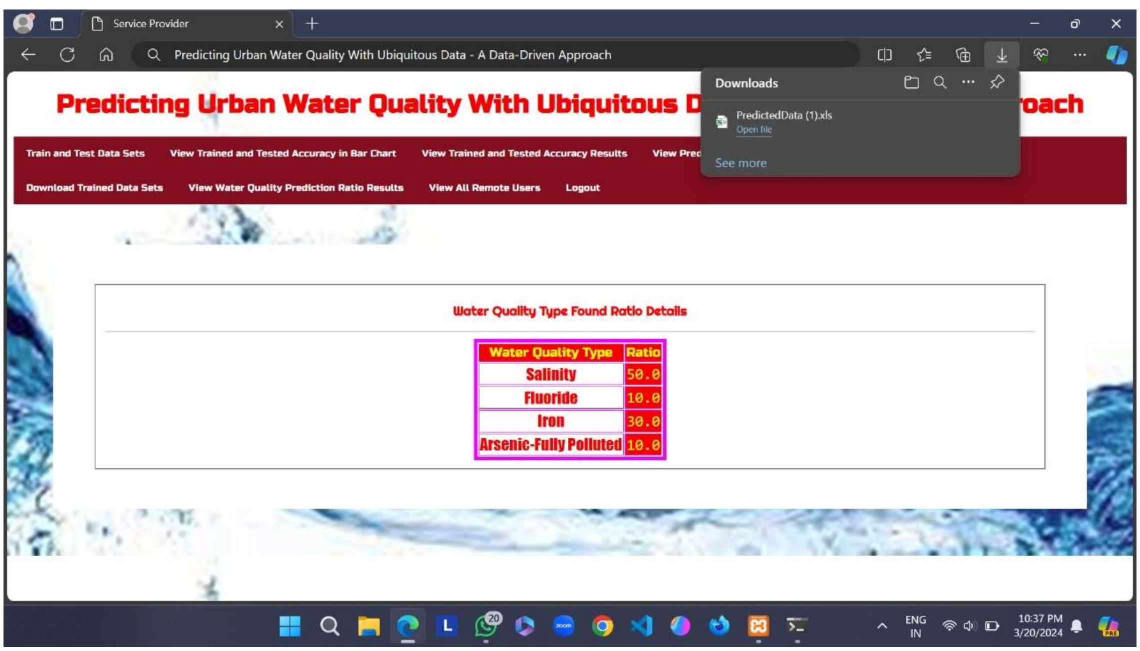
Screenshot 5.6: View Predicted Water Quality Type



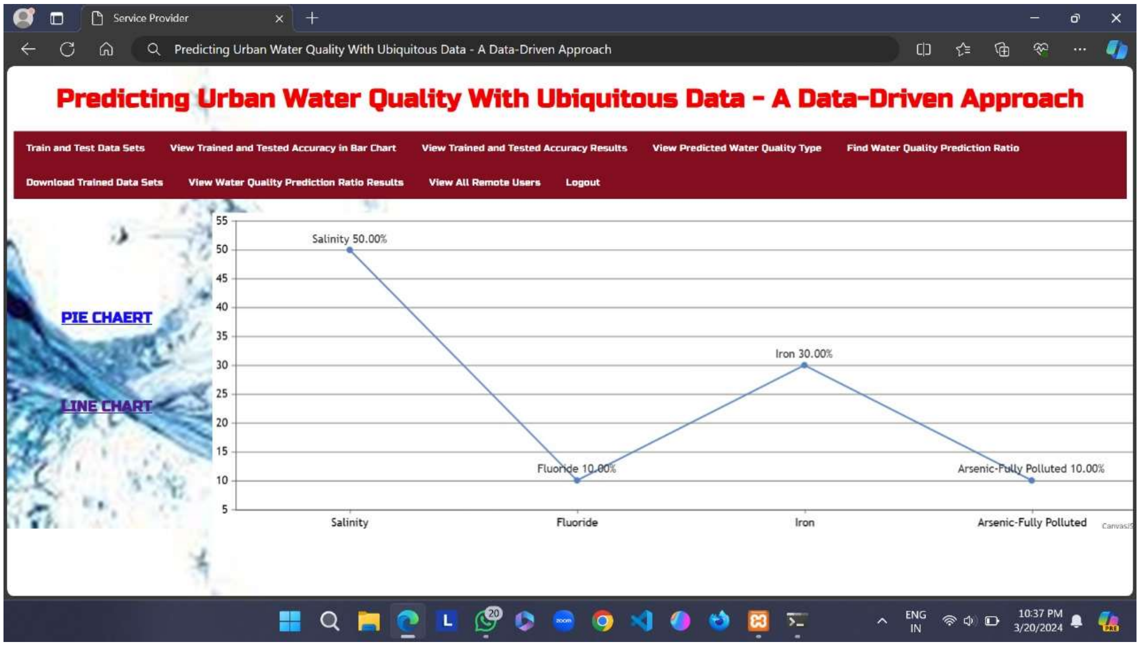
Screenshot 5.7: View Water Quality Prediction Ratio



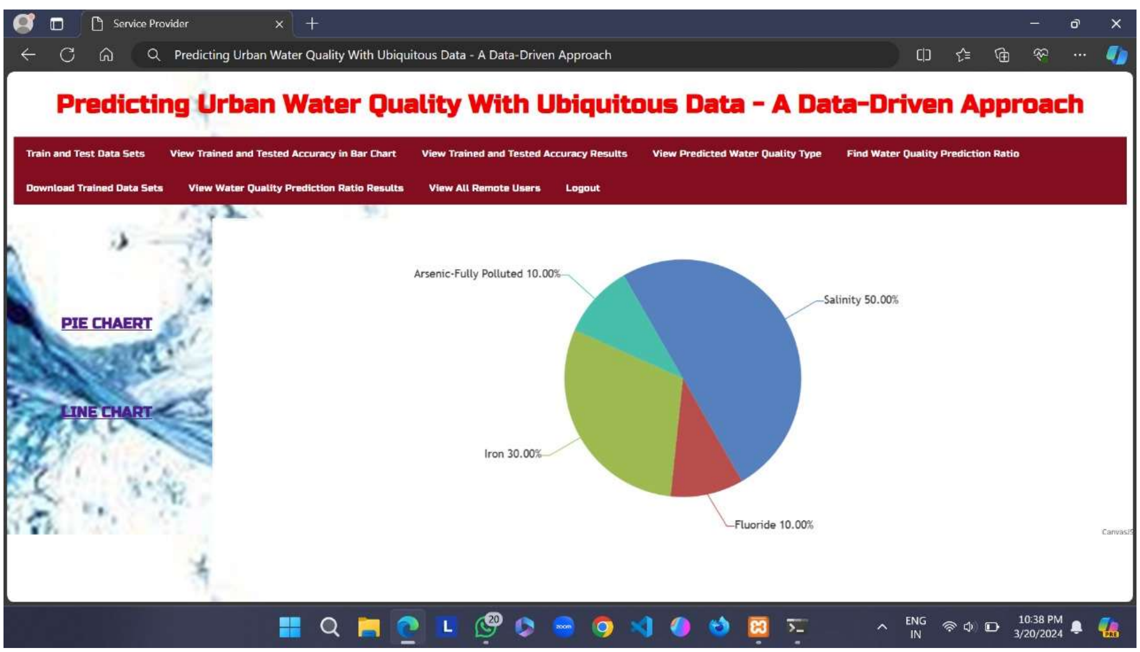
Screenshot 5.8: Download Trained Datasets



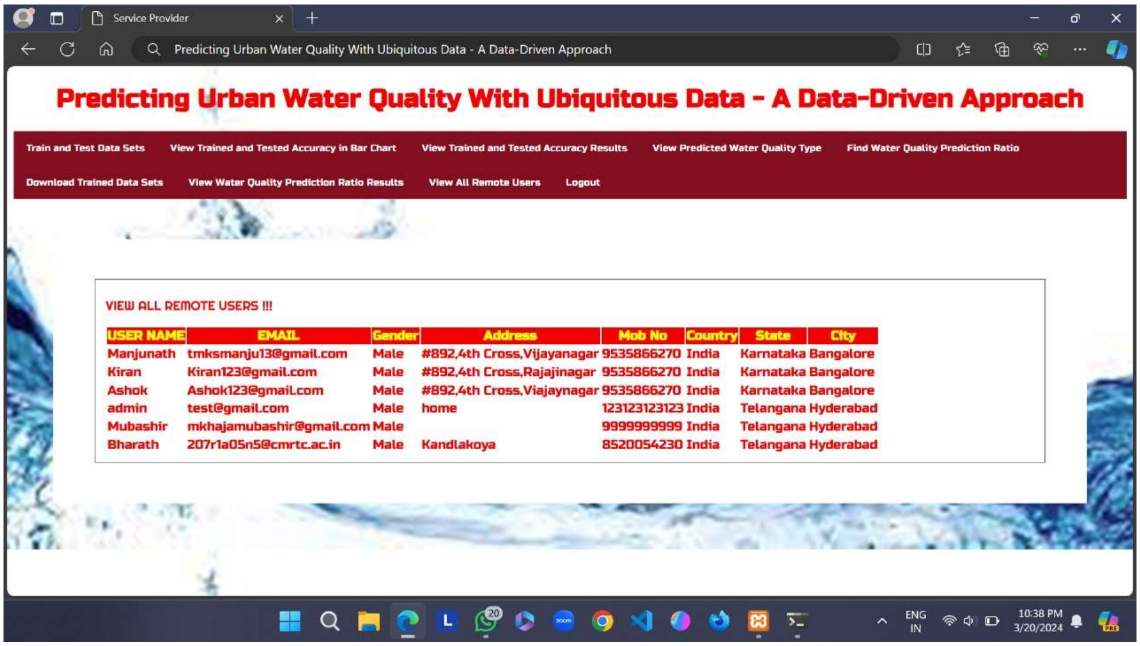
Screenshot 5.9: View Water Quality Prediction Ratio Results in Line Chart



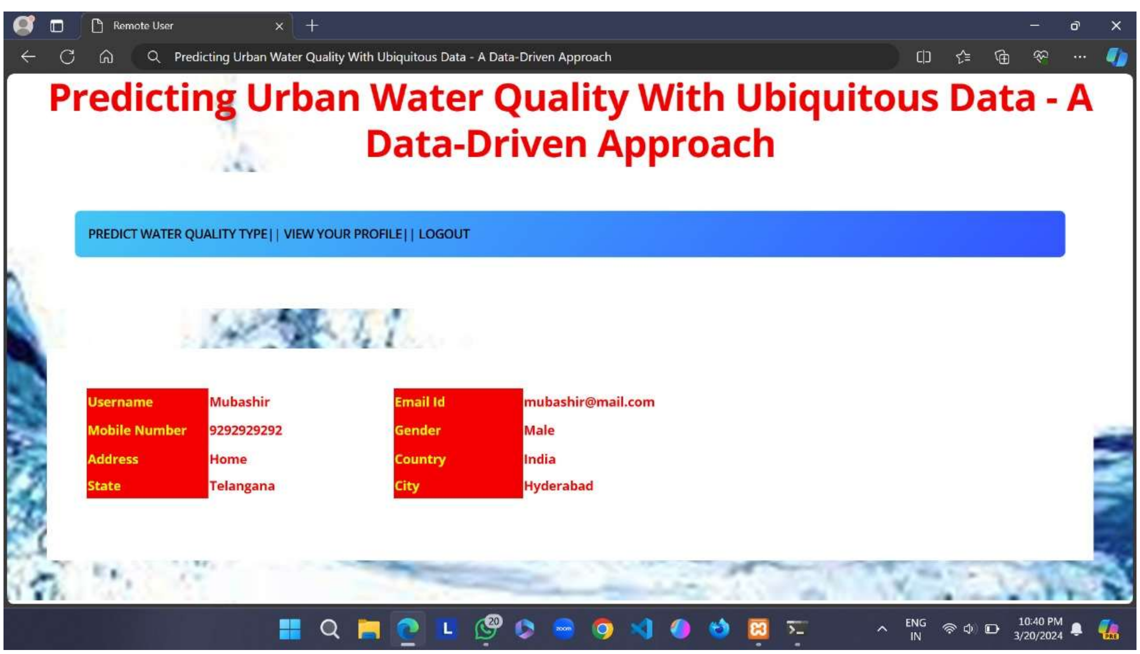
Screenshot 5.10: View Water Quality Prediction Ratio Results in Pie Chart



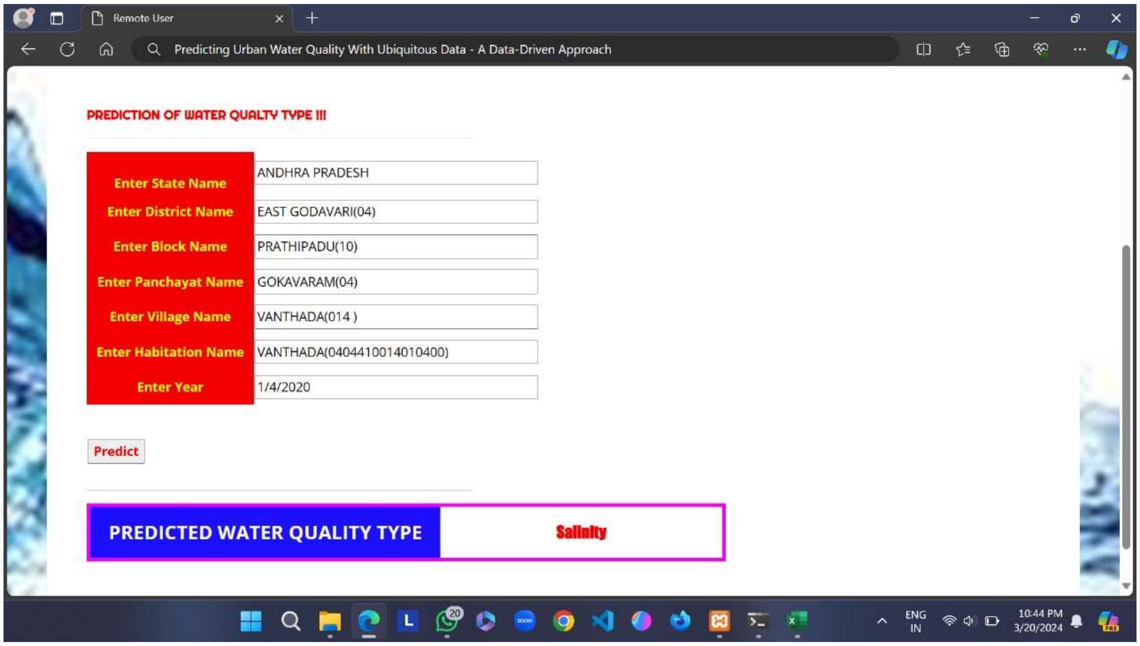
Screenshot 5.11: View all Remote Users



Screenshot 5.12: Remote User View Profile



Screenshot 5.13: Remote User Predicting Water Quality Type



6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

Testing is an integral and indispensable part of the software development and quality assurance process. Its fundamental purpose is to unearth errors and flaws within a work product, ensuring that the final software system is robust, reliable, and capable of meeting user expectations. Testing serves as the systematic process of meticulously scrutinizing every conceivable fault or weakness that might be present in a software product, ranging from individual components and subassemblies to complete assemblies or finished products. At its core, testing functions as a comprehensive evaluation mechanism, verifying the functionality, performance, and reliability of software components and systems.

The overarching objective is to confirm that the software not only adheres to its specified requirements but also aligns with user expectations. Equally important is the assurance that the software does not fail in any unacceptable manner under different operational conditions. To achieve these objectives, testing encompasses a wide array of test types, each addressing specific testing requirements.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Integration testing, like other testing methodologies, is fundamentally designed to uncover errors and inconsistencies. It aims to expose issues that may arise when previously tested components come together and interact. This testing approach endeavors to verify that the combined system behaves as expected, adhering to predefined requirements and meeting user expectations. Integration testing examines different integration points, including APIs, databases, communication protocols, and data transfer mechanisms.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstration that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

Valid Input : Identified classes of valid input must be accepted.

Invalid Input : Identified classes of invalid input must be rejected.

Functions : Identified functions must be exercised.

Output : Identified classes of application output must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases

6.2.4 SYSTEM TEST

System testing is a critical phase in the software testing process, with the primary objective of ensuring that the entire integrated software system aligns with the specified requirements. This type of testing evaluates the software configuration as a whole to verify that it produces known and predictable results. System testing often involves comprehensive scenarios and use cases, mimicking real-world interactions, in order to assess the software's behavior in a holistic manner.

A noteworthy example of system testing is the configuration-oriented system integration test, which is focused on examining how different components and modules are configured to function as a coherent whole. This type of testing is fundamentally grounded in process descriptions and flows, emphasizing pre-determined process links and integration points. It ensures that the software system functions seamlessly when all its components are integrated, ultimately meeting user expectations and fulfilling its intended purpose.

6.2.5 WHITE BOX TESTING

White Box Testing is a sophisticated testing approach that requires the software tester to possess knowledge of the inner workings, structure, and often the programming language of the software being tested. In this method, the tester has insight into the software's internal architecture, allowing them to delve into the logic and algorithms that drive its functionality. White Box Testing is typically employed to scrutinize areas of the software that are not easily accessible through black box testing, providing a more detailed and in-depth assessment of its internal mechanisms. This form of testing is especially useful for identifying coding errors, logical flaws, and security vulnerabilities within the software. It aids in uncovering issues related to the software's control flow, data flow, and overall code quality. White Box Testing aims to improve the structural quality of the software by ensuring that it adheres to coding standards, meets design specifications, and performs as intended at the code level.

6.2.6 BLACK BOX TESTING

Black Box Testing stands in contrast to White Box Testing, as it involves testing the software without any knowledge of its inner workings, structure, or programming language. Testers approach the software as a "black box," focusing solely on its external behavior and functionality. This method does not require an understanding of the software's code or internal logic; instead, it evaluates how the software performs based on the inputs provided and the outputs it generates. Black Box Testing is conducted without considering the software's underlying mechanisms, making it ideal for assessing the software from an end-user perspective.

Test cases are designed based on defined specifications or requirements documents, and the test focuses on validating that the software functions as expected and complies with the documented criteria. This approach is valuable for validating user- facing features, usability, and overall system behavior. In essence, Black Box Testing is centered on evaluating what the software does rather than how it accomplishes it, making it an essential part of ensuring that the software meets user expectations and operates as intended.

6.3 TEST CASES

6.3.1 CLASSIFICATION

S.No	Test Type	Test Objective	Test Case Description	Expected Outcome
1	Registration	Verify user registration functionality	Attempt to register a new user with valid credentials	User is successfully registered and can log in to access the system
2	Login	Verify user login functionality	Attempt to log in with correct username and password	User is successfully logged in and directed to the system dashboard
3	Dataset Submission	Verify dataset submission functionality	Upload a Water Quality dataset file to the system	Dataset is successfully uploaded and stored in the system database
4	Model Training	Verify model training functionality	Initiate training of a machine learning model with provided dataset	Model is trained successfully without errors and ready for testing
5	Results Viewing	Verify result visualization functionality	Initiate prediction of water quality status based on input data	Accuracy metrics are displayed correctly, providing clear insights for analysis

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

The project introduces a groundbreaking data-driven approach designed to forecast future water quality across multiple stations, offering a versatile solution that extends beyond the realm of urban water quality prediction to encompass a wide array of multi-location-based forecasting tasks. By meticulously identifying and incorporating a range of pertinent features, including points of interest, intricate pipe networks, temporal variations, meteorological conditions, and water hydraulic characteristics, the methodology not only significantly enhances water quality prediction accuracy but also exhibits potential for application in broader urban contexts.

Through rigorous evaluation utilizing real-world datasets sourced from diverse regions across various states in India, the approach convincingly outperforms conventional baselines such as ARMA, Kalman filter, and ANN, unveiling compelling insights poised to enrich urban living standards and foster societal well-being.

7.2 FUTURE SCOPE

Integration of Advanced Machine Learning Techniques: As machine learning algorithms continue to evolve, future research can explore integrating more advanced techniques such as deep learning, ensemble methods, and reinforcement learning to further improve the accuracy and robustness of water quality prediction models.

Incorporation of Big Data and IoT: With the rapid increase of IoT devices and the availability of big data, future studies can use these technologies to gather more comprehensive and real-time data on various environmental factors affecting water quality. This could include data from sensors deployed in water bodies, smart infrastructure, and even citizen science initiatives. Integrating such diverse datasets can enhance the predictive capabilities of models.

Spatial and Temporal Scalability: While the current approach considers spatial and temporal factors, there is potential to further refine the modeling techniques to account for scalability across different geographical regions and time scales.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

[1] W.H. Organization Guidelines for drinking-water quality, 2004, vol. 3.

[2] L. A. Rossman, R. M. Clark, and W. M. Grayman, “Modeling chlorine residuals in drinking water distribution systems,” Journal of Environmental Engineering, vol. 120, no. 4, pp. 803–820, 1994.

[3] Y. Zheng, “Methodologies for cross-domain data fusion: An overview,” IEEE Transactions on Big Data, vol. 1, no. 1, pp. 16–34, 2015.

[4] Y. Zheng, H. Zhang, and Y. Yu, “Detecting collective anomalies from multiple spatiotemporal datasets across different domains,” 2015. [6] Y. Liu, Y. Zheng, Y. Liang, S. Liu, and D. S. Rosenblum, “Urban water quality prediction based on multi-task multi-view learning,” in Proceedings of the International Joint Conference on Artificial Intelligence, 2016.

[5] H. Cohen, “Free chlorine testing,” http://www.cdc.gov/safewater/chlorineresidual-testing.html, 2014, accessed on 5 August 2016.

[6] P. Castro and M. Neves, “Chlorine decay in water distribution systems case study–lousada network,” Electronic Journal of Environmental, Agricultural and Food Chemistry, vol. 2, no. 2, pp. 261–266, 2003.

[7] W. M. Grayman, R. M. Clark, and R. M. Males, “Modeling distribution system water quality: dynamic approach,” Journal of Water Resources Planning and Management, vol. 114, no. 3, pp. 295–312, 1988.S. A. Burney and H. Tariq. K-means cluster analysis for image segmentation. International Journal of Computer Applications, 96(4),2014.

[8] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, and Y. Mu, “Piecewise linear approximation of streaming time series data with max-error guarantees,” in Proceedings of the IEEE International Conference on Data Engineering, 2015, pp. 173–184.

8.2 GITHUB LINK

https://github.com/immubashir/predicting-urban-water-quality-with-ubiquitous-data-a-data-drive-approach