A Project Report

On

Water Quality Prediction Using Long Short Term Memory with Deep Deterministic Policy Gradient

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

Nalabothu Naveena	(21NN1A0544)
Battineni Sravya	(21NN1A0505)
Guttula V.S. Sai Lakshmi	(21NN1A0520)
Kasireddy Siva Manogna	(21NN1A0528)
Vissamsetti Navya Sree	(22NN5A0506)

Under the Esteemed Guidance of

Dr. V. Lakshman Narayana

Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIGNAN’S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN

PEDAPALAKALURU, GUNTUR-522005

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada.)

2021-2025

VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN

PEDAPALAKALURU ROAD, GUNTUR-522005

(Approved by AICTE &Affiliated to JNTUK, Kakinada)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this project report entitled "**Water Quality Prediction Using Long Short Term Memory With Deep Deterministic Policy Gradient**" is a bonafide record of work carried out by **Nalabothu Naveena(21NN1A0544), Battineni Sravya (21NN1A0505), Guttula V. S. Sai Lakshmi (21NN1A0520), Kasireddy Siva Manogna (21NN1A0528), Vissamsetti Navya Sree(22NN5A0506)** under the guidance of and supervision of Dr. V. Lakshman Narayana in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of **VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN, GUNTUR**.

Project Guide	**Head of the Department**
Dr. V. Lakshman Narayana	**Dr. V. Lakshman Narayana**
Professor	

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work described in this project work, entitled "**Water Quality Prediction Using Long Short Term Memory with deep Deterministic Policy Gradient**" which is submitted by us in partial fulfilment for the award of **Bachelor of Technology** in the Department of **Computer Science and Engineering** to the **Vignan's Nirula Institute of Technology and Science for Women**, affiliated to Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, is the result of work done by us under the guidance of **Dr. V. Lakshman Narayana** Professor.

The work is original and has not been submitted for any Degree/ Diploma of this or any other university.

Place: Guntur

Date:

PROJECT ASSOCIATES

Nalabothu Navvena	(21NN1A0544)
Battineni Sravya	(21NN1A0505)
Guttula V.S. Sai Lakshmi	(21NN1A0520)
Kasireddy Siva Manogna	(21NN1A0528)
Vissamsetti Navya Sree	(22NN5A0506)

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our beloved principal **Dr. P. Radhika** for giving a chance to study in our esteemed institution and providing us all the required resources.

We would like to thank **Dr. V. Lakshman Narayana**, **Professor, Head of the Department of Computer science and Engineering**, for his extended and continuous support, valuable guidance and timely advices in the completion of this project thesis.

We wish to express our profound sense of sincere gratitude to our Project Guide **Dr. V. Lakshman Narayana, Professor, Department of Computer Science and Engineering**, without whose help, guidance and motivation this project thesis could not have been completed the project successfully.

We also thank all the faculty of the Department of Computer Science and Engineering for their help and guidance of numerous occasions, which has given us the cogency to build-up adamant aspiration over the completion of our project thesis.

Finally, we thank one and all who directly or indirectly helped us to complete our project thesis successfully.

PROJECT ASSOCIATES

Nalabothu Naveena	(21NN1A0544)
Battineni Sravya	(21NN1A0505)
Guttula V.S. Sai Lakshmi	(21NN1A0520)
Kasireddy Siva Manogna	(21NN1A0528)
Vissamsetti Navya Sree	(22NN5A0506)

WATER QUALITY PREDICTION USING LONG SHORT TERM MEMORY WITH DEEP DETERMINISTIC POLICY GRADIENT

ABSTRACT

Water quality prediction is crucial for ensuring environmental sustainability, public health, and industrial applications. This research introduces a water quality prediction model based on Long Short-Term Memory (LSTM) and Deep Deterministic Policy Gradient (DDPG), leveraging reinforcement learning to enhance predictive accuracy. The model analyzes key water quality parameters, including pH, Dissolved Oxygen (DO), Conductivity, turbidity, and temperature, to determine the suitability of water for human consumption. The model leverages LSTM networks to capture temporal dependencies in water quality data, while DDPG, a reinforcement learning algorithm, optimizes the decision-making process by continuously refining predictions. The LSTM-DDPG model processes the data, extracts relevant features, and generates a prediction indicating the water's quality status. The system provides outputs in an easy- to-understand format, helping users assess whether the water is safe for various applications, including agriculture, drinking water supply, industrial usage, and environmental monitoring.

TABLE OF CONTENTS

Chapter No	Name of the Chapter	Page No
Chapter 1 Introduction		1
1.1	Introduction	1
1.2	Understanding the Significance of Variables	2
1.3	Role of Deep Learning in water quality prediction	3
1.4	Challenges in water quality monitoring	3
1.5	Benefits of using deep learning for water quality prediction	4
Chapter 2 Literature Survey		5
2.1	Literature Review	5
2.2	Gaps identified from the Literature Review	12
Chapter 3 System Analysis		13
3.1	System Analysis	13
3.2	Problem Definition	14
3.3	Existing System	14
3.4	Proposed System	16
3.5	Feasibility Study	19
Chapter 4 Requirements Specifications		20
4.1	Purpose, Scope, Definition	20
4.1.1	Purpose	20
4.1.2	Scope	20
4.1.3	Definition	20
4.2	Requirement Analysis	20
4.2.1	Functional Requirements	21
4.2.2	User Requirements Analysis	21
4.2.3	Non-Functional Requirements	21
4.3	System Requirements	22
4.3.1	Software Requirements	22
4.3.2	Hardware Requirements	23

Chapter 5 System Design 24

5.1 System Architecture 24

5.2 Modules 24

5.3 Design Overview 26

5.4 UML Diagrams 28

5.4.1 Use case Diagram 28

5.4.2 Activity Diagram 29

5.4.3 Sequence Diagram 30

5.4.4 Component Diagram 30

5.4.5 Deployment Diagram 31

5.5 Algorithm for Proposed Model LSTM with DDPG 31

Chapter 6 Implementation 33

6.1 Steps for Implementation 33

6.2 Coding 40

Chapter 7 Testing 48

7.1 Testing 48

7.2 Types Of Tests 48

7.2.1 Unit Testing 50

7.2.2 Integration Testing 51

7.2.3 Functional Testing 51

7.2.4 System Testing 51

7.2.5 White Box Testing 52

7.2.6 Black Box Testing 53

7.3 Steps 54

Chapter 8 Results And Discussions 56

8.1 User Interfaces 56

8.2 Experimental Results 58

8.2.1Performance of Existing and Proposed Models. 58

8.2.2 Confusion Matrix of Existing and Proposed models 60

Chapter 9 Conclusion And Future Work 62

Chapter 10 Bibliography 63

LIST OF FIGURES

S.No	Fig No	Description of Figure	Page No
1	1.1	Water Monitoring	1
2	1.2	Using water for Human consumption	3
3	5.1	System Architecture	24
4	5.2	LSTM-DDPG process	25
5	5.3	Types and Categories of UML Diagram	26
6	5.4.1	Use Case Diagram	27
7	5.4.2	Activity Diagram	28
8	5.4.3	Sequence Diagram	30
9	5.4.4	Component Diagram	30
10	5.4.5	Deployment Diagram	31
11	7.1.1	User Interface	54
12	7.1.2	Enter values	55
13	7.1.3	Predict output	55
14	8.1.2	User Interface	56
15	8.1.2	Result Page1	57
16	8.1.3	Result Page2	57
17	8.2.1	Accuracy	59
18	8.2.2	Precision	59
19	8.2.3	Recall	59
20	8.2.4	F1-Score	59
21	8.3.1	Confusion Matrix of LSTM-DDPG	60
22	8.3.2	Confusion Matrix of RNN	60
23	8.3.3	Confusion Matrix of DNN	60
24	8.3.4	Confusion Matrix of LSTM	60

LIST OF TABLES

S.No	Table No	Table Description	Page No
1	2.1	Literature Survey	8

LIST OF ACRONYMS

Acronyms	Definition
LSTM	Long Short-Term Memory
DDPG	Deep Deterministic Policy Gradient
DO	Dissolved Oxygen
PH	Potential of Hydrogen
TDS	Total Dissolved Solid
DL	Deep Learning
RL	Reinforcement Learning
WQP	Water Quality Prediction
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Introduction

Water quality is a crucial factor in maintaining a healthy ecosystem and ensuring safe consumption. With the increasing levels of industrialization and pollution, monitoring and predicting water quality has become essential for public health and environmental sustainability [1]. Traditionally, water quality assessment relies on laboratory testing, which is often time-consuming and expensive [2]. However, advancements in Deep Learning have enabled the development of predictive models that can analyse water quality efficiently and accurately.

This project, "Water Quality Prediction Using LSTM with DDPG in Deep Learning," aims to build a robust predictive system that leverages LSTM networks and DDPG reinforcement learning[3]. LSTM models are effective for time-series forecasting, while DDPG enhances decision-making by optimizing model performance through continuous learning. The proposed approach integrates various water quality parameters such as pH, Dissolved Oxygen (DO), Turbidity, and Conductivity to make real-time predictions[4].

Fig 1.1: Water Monitoring

By implementing this deep learning-based predictive model, this project seeks to automate water quality analysis, minimize manual errors, and improve the efficiency of monitoring water resources[5]. The insights generated from this system can aid environmental agencies, industries, and researchers in taking proactive measures for water conservation and pollution control[6]**.**

Fig 1.2: Using Water for Farming

1.2 Understanding the Significance of Water Quality Parameters

Water quality assessment involves analysing various physicochemical, biological,and environmental parameters to determine whether water is safe for drinking, irrigation, and industrial use[7]. The key parameters considered in this study are:

- **pH Level**: Indicates the acidity or alkalinity of water, affecting aquatic life and human consumption.
- **Dissolved Oxygen (DO):** Essential for aquatic organisms; lower levels can indicate pollution.
- **Turbidity**: Measures water clarity; high turbidity levels suggest contamination from sediments or pollutants.
- **Conductivity**: Conductivity in water measures its ability to conduct electricity, influenced by dissolved salts and minerals. Higher conductivity indicates more dissolved ions, often due to pollution or mineral-rich water sources.

- **Total Dissolved Solids (TDS):** Determines the presence of minerals, salts, and metals in water.

- **Temperature**: Affects biological and chemical processes in water ecosystems. Accurate prediction of these parameters allows early detection of contamination, helping authorities implement corrective actions to maintain safe and sustainable water resources.

1.3 Role of Deep Learning in Water Quality Prediction

Deep Learning has revolutionized predictive modelling by enabling automated feature extraction, pattern recognition, and high-accuracy forecasting. In this project, we use[8]:

1.3.1 LSTM Networks:

Specialized for handling time-series data, allowing the model to learn long-term dependencies between historical water quality data and future trends.

1.3.2 DDPG Algorithm:

A reinforcement learning technique that continuously improves predictions by optimizing decision-making policies for enhanced model performance.

By combining these approaches, we can develop an intelligent system capable of learning from past trends and adapting to new environmental changes, making water quality prediction more reliable and efficient[9].

1.4 Challenges in Water Quality Monitoring

Water quality monitoring faces several challenges, including[10]:

1.4.1 Time-consuming and Costly Laboratory Testing: Traditional methods require significant resources and skilled personnel.

1.4.2 Data Inconsistencies: Variability in data collection methods leads to inconsistencies in measurement accuracy.

1.4.3 Environmental Factors: Sudden changes in temperature, pollution sources, or rainfall patterns can impact predictions.

1.4.4 Need for Real-Time Analysis: Delay in detecting contaminants can lead to severe health and environmental risks.

1.5 Benefits of Using Deep Learning for Water Quality Prediction

Implementing a deep learning-based system offers several advantages:

1.5.1 Automated Predictions: Reduces reliance on manual testing by providing real-time water quality assessments.

1.5.2 High Accuracy: LSTM models can analyze complex patterns, leading to precise predictions.

1.5.3 Early Warning System: Helps authorities take preventive measures before contamination reaches dangerous levels.

1.5.4 Resource Optimization: Minimizes costs and time spent on laboratory testing.

1.5.5 Scalability: Can be applied across different water bodies, from small reservoirs to large industrial water systems.

CHAPTER 2
LITERATURE SURVEY

CHAPTER 2
LITERATURE SURVEY

Nitzan Farhi et al ..,[1] proposed Long Short-Term Memory (LSTM) neural network. This study proposes a machine learning-based approach using Long Short-Term Memory (LSTM) networks to predict ammonia (NH_4^+) and nitrate (NO_3^-) concentrations in wastewater treatment plants (WWTP). It Prevents excessive nitrogen discharge, reducing eutrophication risks. And in this project have demerits also Prevents excessive nitrogen discharge, reducing eutrophication risks.

Jixuan Chen et al.., [2] proposed LTSF-Linear model. This investigation aims to enhance the accuracy of water quality prediction, considering the temporal characteristics, variability, and complex nature of water quality data. The findings contribute to better water resource management, pollution prevention, and ecological restoration. The study mentions the challenge of dealing with multiple influencing factors in water quality prediction, which may limit model adaptability to diverse environments.

Mohammad Ehteram et al..,[3] proposed Convolutional neural network (CNN). It provides CNN for water quality index prediction. It allows the model to capture both spatial and temporal patterns, making it effective at analyzing complex water quality data with varying characteristics. It can be very powerful, they may lack transparency and interpret ability in comparison to simpler models making it harder to understand the exact reasoning behind specific predictions.

Ioannis Partalas et al.., [4] Studied Deep learning models. This article explores the potential of ensemble learning techniques to raise the precision and dependability of water quality assessments derived from remote sensing is examined in this article. More sensors and a wider geographic coverage area can be added to the system as it grows. Easy extension is made possible by the measuring stations' and the central data collection station's modular designs. The quality of the incoming data has a significant impact on machine learning models' performance. If problems such as outliers or missing variables are not correctly handled, the accuracy and dependability of the model may suffer.

Donthula Mamatha et al..,[5] Studied Convolution Neural Network. This research focuses on using deep learning, specifically the VGG19 convolutional neural network, for water quality classification. A channel-wise attention gate is integrated to enhance feature extraction from water surface images. It approach includes preprocessing steps such as background elimination, removal of non-essential features, image enhancement, and noise removal to improve classification accuracy. The accuracy of the model heavily depends on the quality and diversity of the training dataset. Inconsistent or biased data can affect performance.

Bhagavathi Perumal et al..,[6] proposed Long Short-Term Memory (LSTM) Model. LSTM model predicts waste water treatment quality.It improve the accuracy of water quality predictions, especially in the presence of non-point source (NPS) pollution.It need regular updates with fresh data to maintain its accuracy and effectiveness. Without proper maintenance, the predictions may become outdated.

Archana Solanki et al..,[7] Proposed Deep Learning Techniques. Deep learning, including unsupervised learning methods like denoising autoencoders and deep belief networks, provides great prediction accuracy. Because of their capacity to manage data fluctuation, these techniques are reliable for practical uses. Additionally, using artificial neural networks (ANN) and other machine learning models for predictive analysis aids in the early detection of declining water quality. Although the work focuses on ANN and deep learning models, hybrid approaches that could further increase prediction accuracy are not thoroughly explore.

Sathya Preiya V.M et al..,[8] Studied Linear Scaling Normalization (LSN). Accurate temporal forecasts of the water quality index (WQI) and multi-level classification are made possible by it. Improved predictive performance results from the addition of hyper parameter adjustment via the Grasshopper Optimization Algorithm. Due to the intricacy of hyper-parameter tuning, especially with GOA, more model optimization knowledge is required, which could make it more difficult for non experts to use. - A promising approach to water quality evaluation, the WQIPC Hyper-parameter Tuning, Deep Learning for Water Quality Index (WQI) Prediction & Classification Simulation & Performance Analysis (GOA), which further increases model efficience.

Yunjeong Im et al..,[9] Proposed Long Short-Term Memory (LSTM). Gated Recurrent Units (GRU), Sample Convolution and Interaction Networks (SCINet). As evidenced by its reported average forecast accuracy of 98.78% and maximum accuracy of 99.98%, the system can successfully manage the uncertainty and ongoing changes in water quality. While the time series cross-validation method is helpful for validating models, it might not fully account for outliers or unexpected changes in the data, which could result in predictions that are not correct in extreme situations.

K. P. RASHEED ABDUL HAQ et al..,[10] studied Hybrid deep learning (DL) models. Convolutional Neural Network (CNN) with the Long Short-Term M0emory (LSTM) and Gated Recurrent Unit (GRU) for aquaculture WQP. The models can successfully capture long-term dependencies in changes in water quality by integrating LSTM or GRU, which is essential for precise forecasting. Overfitting can result from a lack of data or noise, which compromises the accuracy of predictions. The intricacy of the model architecture is another drawback, requiring substantial computational resources for training and meticulous hyper parameter adjustment, which makes it less practical for small-scale aquaculture farms with inadequate infrastructure.

Table 2.1. Literature Survey

S.No	Author	Methodology	Merits	Demerits
1	Nitzan Farhi, Et al…	Long- Short Term Memory (LSTM)	Early detection of nutrient imbalances and pollution prevention are made possible by the high accuracy of the suggested LSTM-based waste water treatment prediction model, which has 99% accuracy for ammonia and 90% accuracy for nitrate.	High computational complexity and reliance on real-time, high-quality data. Operators that want transparency face difficulties due to its black-box character, which makes interpretation challenging.

2	Mohammd Ehteram, Et al..	Convolutional Neural Network (CNN),	Using deep learning and decision tree approaches, the proposed CNN-CRNN- M5T model improves the precision and effectiveness of Water Quality Index (WQI) prediction. - The model efficiently captures complex correlations in water quality metrics by combining CNN for feature extraction, Clockwork RNN for sequential pattern recognition, and M5T for decision-making.	Including the need for high-quality data and a high level of computing complexity in order to make accurate predictions. - Combining several algorithms lengthens processing times and necessitates technical know-how for setup and upkeep.
3	Bhagavati Perumal, Et al..	Deep learning	By managing nonlinear and nonstationary data, deep neural networks, convolutional neural networks(CNNs), support vector regression (SVR), and hybrid techniques like LSTM-GWO-FSO greatly increase prediction accuracy. These models lessen reliance on traditional in- situ techniques by processing bigdata sets,spotting trends and maximizing real-time monitoring using remote sensing.	Large volumes of high- quality training data are necessary for AI and ML techniques, yet inaccurate data gathering can produce predictions that are not trustworthy. Despite being more transparent, mechanistic models are frequently intricate,computation ally demanding, and difficult to calibrate, especially in real- time applications.

4	Archana Solanki, Et al..	Deep Learning Techniques	Deep learning, including unsupervised learning methods like denoising autoencoders and deep belief networks, provides great prediction accuracy. Because of their capacity to manage data fluctuation, these techniques are reliable for practical uses. .	Even though unsupervised learning is a strong technique, real- time applications may not always be able to support the massive datasets and processing resources that it requires. Furthermore, although the work focuses on ANN and deep learning models, hybrid approaches that could further increase prediction accuracy are not thoroughly explored.
5	Sathya Preiya V. M. Et al...	Long Short- Term Memory (LSTM) Model,	Accurate temporal forecasts of the water quality index (WQI) and multi- level classification are made possible by it. Improved predictive performance results from the addition of hyper parameter adjustment via the Grasshopper Optimization Algorithm(GOA), which further increases model efficiency.	Due to the intricacy of hyper-parameter tuning, especially with GOA, more model optimization knowledge is required, which could make it more difficult for non- experts to use. A promising approach to water quality evaluation, WQIPC-HTDL method overcomes many of the shortcomings of traditional approaches while striking a balance between accuracy and efficiency.

6	YunjeongI m , Et al….	Long Short-Term Memory (LSTM)	As evidenced by its reported average forecast accuracy of 98.78% and maximum accuracy of 99.98%, the system can Successfully manage the uncertainty and ongoing changes in water quality. In addition to offering faster, more accurate, and scalable forecasts than conventional techniques like the ARIMA model, this predictive approach also increases public health safety and reduces concerns associated with water quality nationwide.	While the time series cross-validation method is helpful for validating models, it might not fully account for outliers or unexpected changes in the data, which could result in predictions that are not correct in extreme situations. Additionally, although deep learning methods increase forecast accuracy, they necessitate modeling, training, and fine- tuning skills, which may restrict their applicability to all areas or smaller water supply systems.
7	K.P.RASH ABDULHAQ ,Et al..,	Convolutional Neural Network (CNN)	By effectively extracting important features from high-dimensional water quality data, the CNN component greatly lowers complexity and enhances the model's capacity for generalization. The models can successfully capture long-term dependencies in changes in water quality by integrating LSTM or GRU.	Overfitting due to data scarcity or noise, and the complexity of the model architecture, which makes it less practical for small-scale aquaculture farms with limited infrastructure. CNN also improves feature extraction, but if max-pooling is too aggressive, it may lose important temporal information, which could reduce prediction accuracy.

8	RanraL, Et al..	Fuzzy Water Pollution Index (FWPI)	The FWPI shows more consistency and dependability in assessing water quality levels than other assessment techniques like fuzzy comprehensive evaluation and the grey relational method.	One drawback is the difficulty of creating a reliable fuzzy inference system, which necessitates meticulous rule- making and indicator weight assignment. Inaccuracies or biases could be introduced into the evaluation findings if the system is not calibrated correctly.
9	Jixuan Chen, Et al..	LTSF-Linear model	The efficiency and simplicity of computation of the suggested methodology. The LTSF-Linear model provides a simplified method that works well even with big datasets, in contrast to deep learning models like LSTM and Informer, which demand substantial processing resources and intricate training procedures.	Despite its efficiency, the linear model might not be as good as more sophisticated deep learning models for identifying highly nonlinear connections in water quality data. The LTSF-Linear model offers a viable, effective, and precise method for predicting water quality, making it a useful instrument for pollution control and sustainable water management.
10	Donta Mamat ha, Et al..	Deep Learning with VGG19 Approach.	Specifically in terms of accuracy and automation. Utilizing deep learning reduces the need for expensive laboratory testing and manual inspections, speeding up and lowering the cost of the process. Preprocessing methods are essential for improving image quality and supply the deep learning model with the best possible input for reliable performance.	The quality and diversity of the data greatly affect how well the model performs. When presented with new data, the model can have trouble generalizing if the training photos don't cover a broad variety of water conditions.

2.2 Gaps identified from the Literature Review

The following are the research gaps identified from the literature review encompassing the work of ten different authors.

- Most deep learning models are black-box in nature, lacking transparency and interpretability for decision-makers.
- High dependency on real-time, clean, and large datasets makes many models impractical in real-world scenarios with noisy or incomplete data.
- Many models demand high computational power and advanced technical skills, limiting their accessibility and scalability.
- Existing solutions often lack generalizability across different geographic locations and water quality conditions.
- Hybrid approaches and optimization techniques are underexplored, despite their potential to enhance model performance.
- Anomalies, outliers, and sudden changes in water quality data are not adequately handled, affecting model robustness.
- Most research focuses on theoretical or simulated environments without real- world validation or deployment.
- There is a lack of user-friendly interfaces or systems, making it difficult for non-experts to adopt these advanced models.

CHAPTER 3

SYSTEM ANALYSIS

CHAPTER 3
SYSTEM ANALYSIS

3.1 System Analysis

Enhancing water quality prediction is crucial for ecological balance, safe drinking water, and efficient environmental monitoring. Conventional approaches to water quality assessment rely on laboratory testing and manual sampling, which can be costly, time-consuming, and unsuitable for large-scale real-time monitoring. To overcome these drawbacks, sophisticated deep learning techniques provide an automated and effective way to predict water quality parameters with high accuracy.

In order to forecast important factors including pH, turbidity, dissolved oxygen, and conductivity, the suggested method primarily analyzes historical and current data on water quality. The system learns from previous trends to predict future changes by using a model that can analyze sequential data. This method increases the accuracy of forecasts by capturing long-term interdependence in changes in water quality. Reinforcement learning methods are used to enhance decision-making in order to increase adaptability even further.

Over time, the model improves forecast accuracy by dynamically adjusting its parameters based on ongoing learning from real-time environmental changes. This makes the system stable and scalable by ensuring that seasonal variations, pollutant sources, and climate fluctuations are taken into consideration. Adaptive learning mechanisms are integrated to ensure that the system continuously improves and generalizes well across various water bodies, locations, and environmental conditions. This approach is implemented by collecting data from sensors and monitoring stations, and then preprocessing the data to remove noise and irrelevant features.

The model then analyzes the data, identifying patterns and relationships that contribute to changes in water quality. The technology helps environmental agencies, water management authorities, and legislators make proactive decisions by offering precise and up-to-date forecasts. Better resource management, water treatment process optimization, and early contamination detection are made possible by this.

Effective water quality monitoring and prediction lessens reliance on conventional testing techniques, supporting sustainable water resource management and guaranteeing access to clean, safe water for a range of uses.

3.2 Problem definition

The problem definition addressed by Predicting water quality is super important to keep people healthy, manage water resources wisely, and protect nature, but the old ways of doing it like collecting water by hand and testing it in a lab are slow, expensive, and take a lot of work. These methods mean we're often too late to stop bad stuff, like pollution spills, from making water unsafe. Water's tricky to figure out because it involves things like acidity (pH), murkiness (turbidity), oxygen levels, toxic metals, and germs, and we need answers fast not days later. With problems like climate change heating things up, farm runoff dirtying rivers, and factories dumping waste, we need smarter, quicker ways to predict water quality using tools that can handle big data from sensors or satellites.
Right now, models like LSTM, RNN, and DNN are trying to help, but they've got issues. LSTM is good at tracking changes over time, like how rain affects a lake, but it struggles to keep up with sudden new problems. RNN wants to follow the order of water data, but it forgets the past too easily, missing slow shifts like pollution building up. DNN can spot complicated patterns, like how factory waste messes with oxygen, but it's not built for tracking changes day by day and needs tons of data to even start. All three have trouble giving fast, clear answers we can trust, especially when water conditions keep shifting. We need these models to fix these gaps and predict water quality better, faster, and without so much hassle.

3.3 Existing System

Deep learning methods, like LSTM, RNN, and DNN, have become a strong choice for improving water quality prediction by tackling large datasets and uncovering complex patterns that traditional tools struggle to catch. DNN rely on lots of layers to find tricky connections between water features like how pH, turbidity, solids, or germs all tie together in a river or pond. For instance, it might notice that high solids and funky conductivity often mean bad water, boosting accuracy when it's fed plenty of labeled data, like a big pile of test results marked "clean" or "dirty." But if it doesn't have enough examples or isn't tuned well, it can overfit making guesses that sound smart but miss the mark. RNN are designed for data that flows in order, like a chain of water tests one day's pH, the next day's turbidity and they're good at spotting short-term changes, like how a rainstorm spikes cloudiness or a spill bumps up germ counts.

However, they're not great at remembering longer shifts, so they might skip how a past drought still affects a lake's oxygen, leaving gaps in the bigger picture. LSTM, a fancier take on RNN, shines by handling time-based data better it follows how water changes step by step, picking up patterns like how factory runoff slowly messes with a stream's quality or how steady rain pushes up solids in a well. It's awesome at guessing what's next because it keeps those trends in mind, but it's not quick to shift gears if something sudden hits, like a chemical leak, and it lacks a way to fine-tune itself on the go. These models plow through messy water quality data fast, pulling out key details like warning signs of pollution way better than old lab methods. Still, they've got their quirks: DNN craves tons of labeled data and stumbles with time trends, RNN forgets too much to catch slow changes, and LSTM, while a pattern pro, can lag when the game changes fast.

Disadvantages of Existing System

When putting these models into practice, there are some issues that need to be resolved. They need tons of computer power, making them slow and costly to run, and they crave lots of clean, labeled data like pH or turbidity tests which is hard to get. They're not fast enough for real-time warnings, like spotting a spill in a river, and they struggle in new places, needing extra work to adjust. Messy data, like wrong germ counts in a pond, throws them off, and you can't easily see how they make guesses, so it's tough to trust or fix them when they're wrong.

High Computational Cost

LSTM, RNN, and DNN need a lot of computer power, which makes them expensive and slow. LSTM takes big machines to track water changes, like oxygen in a river. RNN needs the same to follow test lists, like pH shifts. DNN's layers, linking germs and cloudiness in a pond, use tons of energy too. It's like running a big engine—costly and not so green.

Dependency on Large Datasets

These models need heaps of water data to work, but that's hard to get. LSTM wants lots of examples, like turbidity tests, to guess right. RNN needs a pile too, like solids readings, or it flops. DNN craves even more, like tagged "safe" or "bad" samples for a well, but water's different everywhere, so if data's missing, they all mess up.

Limited Real-Time Processing Capabilities

They’re slow at giving quick answers. LSTM lags tracking a river’s solids fast enough for a spill. RNN can’t keep up with germ jumps in a lake. DNN’s too busy sorting pH and turbidity in a stream to warn right away.

Limited Generalization to New Locations

They don’t work well in new places. LSTM might get a river but not a pond with different dirt. RNN could handle a well but fail on a stream with odd weather. DNN links stuff in one lake but bombs elsewhere unless you redo it all. Each spot’s quirks mean extra work.

Sensitivity to Noisy and Incomplete Data

Bad or missing data throws them off. LSTM stumbles if a river’s oxygen test is wrong. RNN messes up a lake’s pH with gaps. DNN can’t sort a pond’s germs and solids if numbers are funky. They need clean data, or their guesses go wild.

3.4 Proposed System

In the proposed LSTM-DDPG model, a combination of LSTM networks and the DDPG algorithm is employed to enhance the accuracy and reliability of water quality prediction for human consumption. These advanced deep learning and reinforcement learning techniques work together to extract meaningful patterns from time-series water quality data and dynamically improve prediction performance.
LSTM is a specialized recurrent neural network (RNN) designed to handle time-series data, making it ideal for analyzing water quality parameters such as pH, conductivity, turbidity, solids, and temperature. It processes historical data, capturing long-term dependencies and trends, which helps in predicting future fluctuations in water quality. This mechanism allows the model to learn from past data and forecast future water quality levels with high precision, making it more effective than traditional deep learning models like DNN, RNN which lack the ability to dynamically learn from sequential data.
DDPG is a reinforcement learning algorithm that enables the model to dynamically adjust predictions based on real-time feedback. Unlike static machine learning models, DDPG continuously learns from new data, improving decision-making by balancing exploration (trying new prediction strategies) and exploitation (applying learned strategies for

accuracy). This reinforcement learning approach ensures that the model adapts to environmental changes, handling uncertainties such as sudden pollution events or seasonal variations in water quality. Additionally, DDPG employs policy optimization and experience replay to refine its predictions, leading to continuous improvement in accuracy without the need for frequent manual updates.

The combined LSTM-DDPG model works by integrating the strengths of both time-series analysis and adaptive decision-making to predict water quality with high accuracy. The model efficiently processes and predicts water quality by combining historical data analysis with real-time adaptive learning, ensuring highly accurate and dynamic predictions. This high accuracy in water quality prediction is crucial for ensuring reliable decision-making, minimizing risks, and optimizing resource management. The LSTM-DDPG model, with its ability to achieve over 98 accuracy, provides significant advantages across multiple domains, particularly in agriculture, environmental monitoring, and public health.

Advantages of Proposed System

High Prediction Accuracy:

The proposed system achieves 98% accuracy, significantly outperforming traditional models like RNN (44%) and DNN (65%). The combination of LSTM for sequential data processing and DDPG for adaptive learning ensures minimal prediction errors.

Real-Time Adaptability

Unlike static models, DDPG continuously learns from real-time environmental changes, making the system highly responsive to sudden fluctuations in water quality. It adjusts predictions dynamically based on new data inputs, ensuring up-to-date and reliable results.

Long-Term Dependency Learning

LSTM effectively captures long-term trends in water quality by processing historical data, allowing for accurate forecasting of future conditions. This helps in early detection of contamination, pollution events, and seasonal variations.

Efficient Handling of Noisy and Uncertain Data

Traditional models struggle with missing values, outliers, and fluctuating water parameters. LSTM's memory cell structure and DDPG's experience replay mechanism improve robustness, reducing the impact of noise and missing data.

Optimized Water Resource Management

The model helps farmers, industries, and municipal authorities make data-driven decisions regarding water usage. By predicting water quality accurately, it prevents soil degradation, protects crops, and optimizes irrigation planning.

Cost-Effective and Scalable Solution

The LSTM-DDPG system reduces manual monitoring costs by automating predictions through AI and IoT-based smart monitoring. Its ability to continuously improve with new data makes it a scalable and long-term solution for water management.

3.5 Feasibility Study

The feasibility study evaluates whether the LSTM-DDPG model can be effectively implemented for water quality prediction by analyzing its technical, economic, operational. This ensures that the system is viable, efficient, and beneficial for agriculture, public health, and environmental monitoring.

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

Technical Feasibility:

The technical feasibility of the LSTM-DDPG model focuses on its ability to handle large-scale water quality data, process real-time sensor inputs, and provide highly accurate predictions. The model requires high-performance computing resources during training, such as GPUs or cloud-based processing, but once deployed, it can efficiently run on cloud servers. Additionally, the model is compatible with widely used machine learning frameworks like TensorFlow and PyTorch, making it easy to integrate into existing environmental monitoring systems.

Operational Feasibility:

The operational feasibility assesses whether the LSTM-DDPG model can be effectively deployed and used in real-world scenarios. Unlike traditional monitoring methods that require manual sampling and laboratory testing, this model provides automated, continuous predictions without human intervention. The system can be accessed through a user-friendly dashboard, where farmers, environmental agencies, and industrial operators can view real-time water quality insights, contamination alerts, and trend analysis.

Economic Feasibility:

The economic feasibility examines whether the proposed system is cost-effective and provides long-term financial benefits. Implementing LSTM-DDPG requires an initial investment for data collection, hardware setup, and model training, but the operational costs are significantly lower compared to traditional water quality monitoring methods. The model helps reduce manual testing costs, which often require frequent chemical analysis, laboratory testing, and human intervention. Additionally, the high accuracy of predictions prevents economic losses in agriculture, public health, and industrial sectors by ensuring that water contamination issues are detected early.

CHAPTER 4
REQUIREMENTS SPECIFICATIONS

CHAPTER 4
REQUIREMENTS SPECIFICATIONS

4.1 Purpose, Scope, Definition

4.1.1 Purpose

The purpose of the software requirements Specification is the basis for the entire project. It lays the foundation and also framework that every team involved in the development will follow. It is used to provide critical information to multiple teams like development, quality assurance, operations, and maintenance. Software requirements specification is a rigorous assessment of requirements before the more specific system designs and its goal is to reduce later the redesign. It should also provide a realistic basis for estimating product costs, risks, and also schedules.

4.1.2 Scope

The scope is the part of project planning that involves determining and documenting a list of specific project goals, deliveries, features, functions, tasks, deadlines, and ultimately costs. In other words, it is what needs to be achieved and the work that must be done to deliver a project. The Software scope is well defined boundary which encompasses all the activities that are done to develop and deliver the software product. The software scope clearly defines the all functionalities and artifacts to be delivered as a part of the software.

4.1.3 Definition

The Software Requirement Specification is a description of a software system to be developed. It is model of a software system to be developed. It is model after business requirements specification, also known as the stake holder requirements specification.

4.2 Requirement Analysis

The process to gather the software requirements from clients, analyze and document them is known as requirements engineering or requirements analysis. The goal of requirement engineering is to develop and maintain sophisticated and descriptive -System/Software Requirements Specification documents. It is a four steps process generally, which includes:

- ➢ Feasibility Study
- ➢ Requirements Gathering
- ➢ Software Requirements Specification
- ➢ Software Requirements Validation

Project Requirements:

- ➢ Research Papers
- ➢ Water Quality Datasets
- ➢ Computational Resources (GPU, Server, or Cloud)
- ➢ Camera (if using visual analysis)

4.2.1 Functional Requirement Analysis

Functional requirements define what the system must do by outlining the key tasks, actions, and activities necessary for operation. These serve as the foundation for functional analysis and system implementation.

For this project, the system should:

- Collect, process, and analyze water quality data.
- Implement a DDPG with LSTM model for accurate water quality prediction.
- Provide real-time forecasting of water quality parameters.
- Display predictions and historical trends through visual dashboards.
- Allow user input for additional data entry and model retraining.

4.2.2 User Requirements Analysis

User requirements define what end-users expect from the system. These must be quantifiable, relevant, and detailed to ensure usability.

The system should:

- Provide an easy-to-use interface for researchers, environmentalists, and policymakers.
- Deliver clear and interpretable results, including visual graphs and reports.

- Support real-time monitoring and alerts for critical water quality changes.
- Allow integration with external sensor data or APIs for real-world applications.

4.2.3 Non-Functional Requirement Analysis

Non-functional requirements define the overall system characteristics and quality attributes.

Performance:

- Efficient data processing for large datasets.
- Fast response time for prediction generation.
- High throughput (handling multiple predictions simultaneously).
- Optimized data transmission time for real-time applications.

Response Time:

- The system should provide predictions within a short processing time after receiving input data.

Scalability & Utilization:

- The system should be scalable to support increased data volume.
- It must efficiently utilize computational resources (CPU/GPU) for deep learning model execution.

4.3 System Requirements

By meeting these software and hardware requirements, developers can effectively develop, test, and deploy the Water Quality Prediction System using LSTM-DDPG, ensuring compatibility, performance, and scalability across different development and deployment environments. Additionally, adherence to these requirements ensures efficient utilization of resources and facilitates seamless integration of the system into water monitoring workflows, contributing to improved decision-making and early detection of water quality issues.

4.3.1 Software Requirements

Operating System – Windows 10 or 11 Ultimate, Linux (Ubuntu 20.04+), MacOS.

Visual Studio Code: Utilized as the Integrated Development Environment (IDE) for writing and editing code, facilitating collaborative development, and managing project files.

Google Colab: Used for data preprocessing, model training, and testing deep learning and reinforcement learning algorithms in a cloud-based Jupyter notebook environment, leveraging Google's computational resources.

Packages: numpy, pandas, scikit-learn, TensorFlow/Keras, PyTorch.

Python Anywhere: Employed for hosting the web application, providing an online platform for deploying and managing the water quality prediction system.

TensorBoard: Used for monitoring LSTM-DDPG model training performance, visualizing loss trends, and debugging the neural network.

Blockchain (Ethereum / Hyperledger Fabric): Integrated for secure and immutable water quality data storage, ensuring data integrity.

4.3.2 Hardware Requirements

Processor – i3/i5: The system requires a high-performance processor for deep learning computations and reinforcement learning optimization. A standard PC with an Intel Core i3/i5 processor ensures efficient processing during development and deployment.

RAM - Minimum 16GB (32GB+ recommended for training): A minimum of 16GB RAM is required to handle large-scale datasets, reinforcement learning computations, and model inference tasks.

GPU -: A dedicated GPU is essential for accelerating LSTM-DDPG model training and deep learning computations

Hard Disk Drive - Minimum 1TB SSD: Fast storage is necessary to handle large datasets, model weights, and training logs. A minimum of 1TB SSD ensures high-speed read/write operations and efficient data handling.

High-Speed Internet (5G / Fiber Optic): Required for real-time data transmission from IoT sensors to cloud-based processing units.

CHAPTER 5

SYSTEM DESIGN

CHAPTER 5
SYSTEM ANALYSIS

5.1 System Architecture

Identify and select a suitable dataset for water quality prediction. Perform Data Pre-Processing including normalization and feature engineering. Split the pre-processed data into Training and Testing sets. Choose an algorithm LSTM with DDPG for water quality prediction based on its suitability for sequential data processing. Train and test the LSTM-DDPG model then generate the Model. Using that Model, create a Web Application using Django and access the Web Application by giving input parameters to get the predicted water quality as the output.

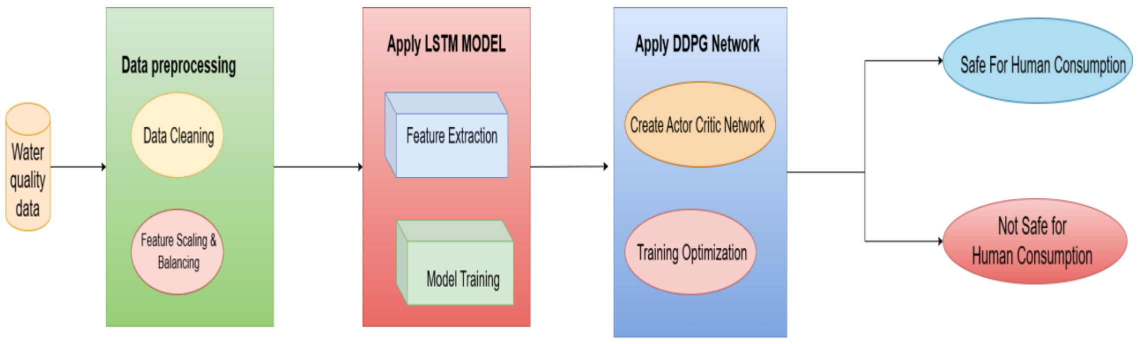


Fig 5.1: System Architecture

5.2 Modules

There are 2 modules used. They are:

1. **LSTM + DDPG as Deep Learning Model**
2. **Flask Framework for User Interface**

5.2.1 LSTM + DDPG Deep Learning Model

Component diagrams are used to display various components of a software system as well as subsystems of a single system. They represent physical components of a system and visualize its structure and organization.

LSTM networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies, making them suitable for sequential data like water quality time series. DDPG is a reinforcement learning algorithm that can optimize decision-making processes.

LSTM Gates

1. Forget Gate
2. Input Gate

3. Cell State Update

4. Output Gate

DDPG utilizes an actor-critic approach where the actor network determines actions based on current states, and the critic network evaluates the quality of those actions.

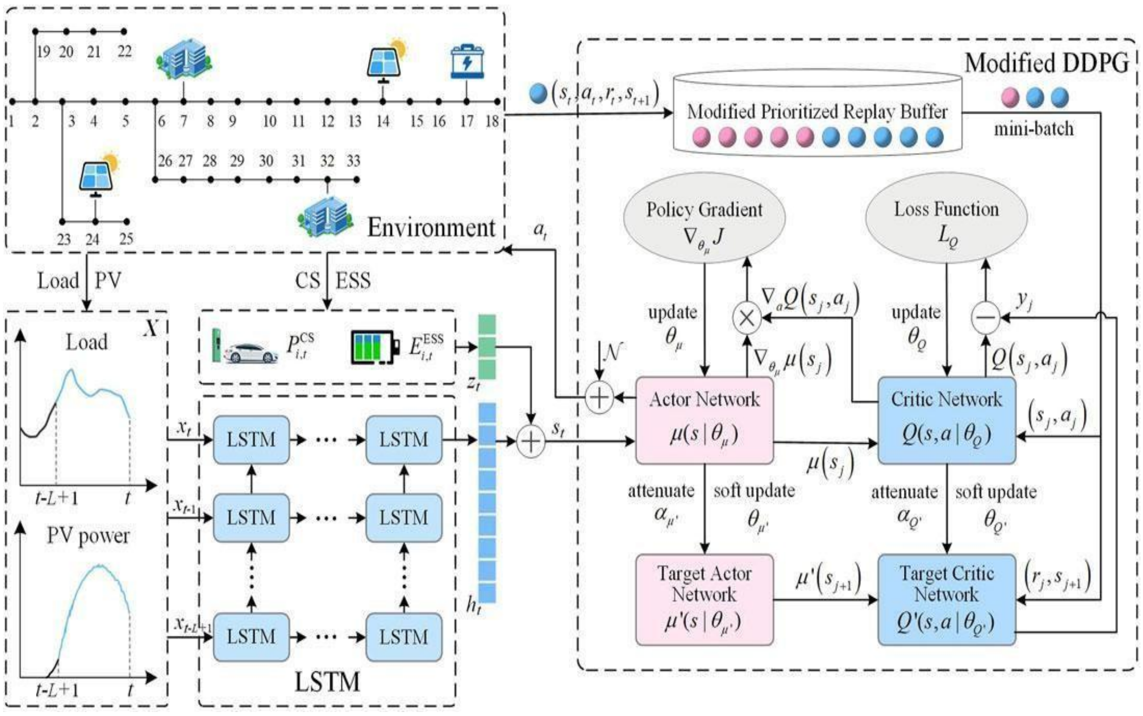


Fig 5.2.1: LSTM-DDPG Process

5.1.1 Flask Framework for User Interface

Flask is a powerful web framework for developing the user interface of a Water Quality Prediction System. It is ideal for building web applications where simplicity and rapid development are essential. Flask is particularly well-suited for integrating machine learning models into web-based interfaces due to its compatibility with Python libraries such as scikit-learn, pandas, and numpy. Flask is used to develop a web-based user interface that allows users to input water quality parameters and receive predictions about water potability. The backend is powered by a pre-trained Deep learning model, which processes the user inputs and provides real-time predictions. Users can enter values for parameters such as pH, turbidity, conductivity, solids. Inputs are passed to a trained DL model which predicts whether the water is safe for human consumption or not. Additionally, Flask provides seamless integration with external databases such as PostgreSQL, MySQL, and SQLite, allowing developers to choose the best storage solution for their application. Flask's

URL routing system allows developers to organize different functionalities efficiently. By mapping specific URLs to different views, Flask ensures that the user experience remains intuitive and well-structured. For example, separate routes can be defined for entering environmental data, viewing historical trends, and accessing real-time predictions.

5.3 Design Overview

The Water Quality Prediction (WQP) model integrates LSTM networks with DDPG reinforcement learning to enhance predictive accuracy. LSTM is utilized to capture temporal dependencies in sequential water quality data, while DDPG optimizes the prediction process by refining model outputs through continuous learning. The system collects real-time and historical water quality parameters such as pH, turbidity, dissolved oxygen (DO), and total dissolved solids (TDS) from IoT-based sensors or datasets. Preprocessing techniques, including normalization and feature selection, ensure data quality before feeding it into the model. The LSTM-based feature extraction module processes time-series data, identifying patterns and anomalies. The DDPG algorithm, with its Actor-Critic architecture, dynamically optimizes water quality predictions by adjusting model parameters based on feedback. A decision support system integrates reinforcement learning rewards to enhance prediction reliability and inform regulatory decisions. The system also includes a visualization and reporting module, generating graphical analyses, anomaly alerts, and interactive dashboards for stakeholders.

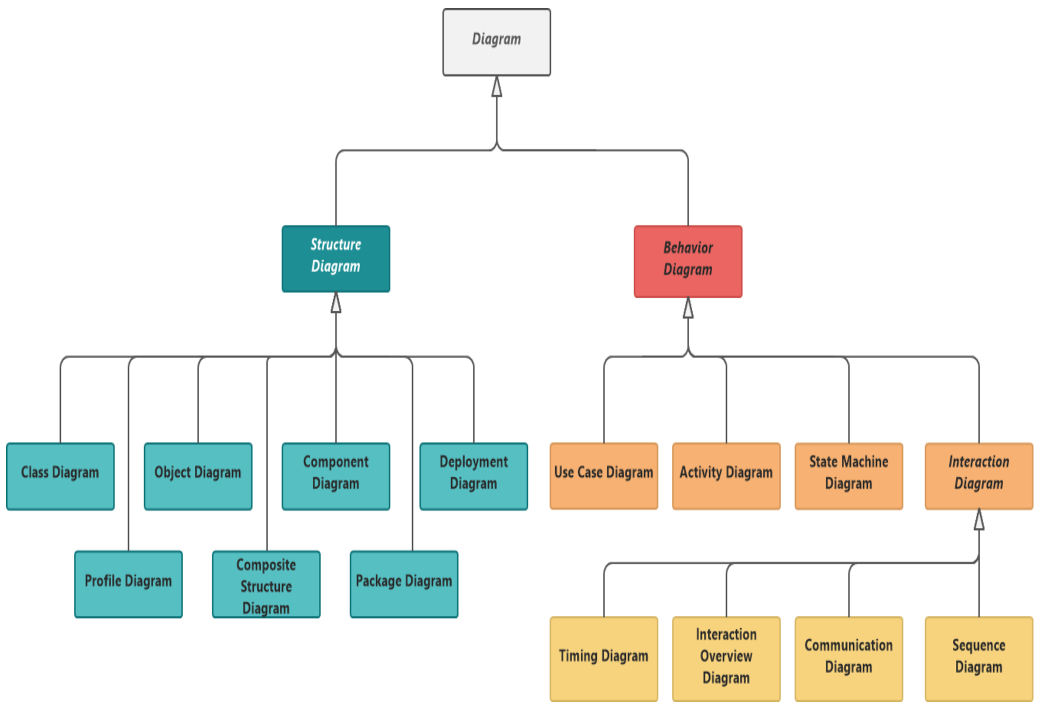


Fig 5.3: Types and categories of UML diagrams

UML diagrams such as use case diagrams, sequence diagrams, and class diagrams represent the system's workflow, illustrating interactions between sensors, AI models, and decision-makers. The use case diagram highlights the roles of sensors, data processors, and users in the system. The sequence diagram details the data flow from collection to prediction and decision-making, while the class diagram defines key system components, including the LSTM-based feature extractor and the DDPG-based optimizer. The UML-based design ensures the system is scalable, adaptable, and easy to integrate with various technologies throughout the software development lifecycle. By leveraging deep learning and reinforcement learning, this model offers a robust approach to real-time water quality monitoring and predictive analysis.

5.4 UML Diagrams

The Unified Modelling Language (UML) is used to specify, visualize, modify, construct, and document the articrafts of an object-oriented software-intensive system under development. UML offers a standard way to visualize a system‘s architectural blueprints, including elements such as:

- Actors
- Business process
- (Logical) Components
- Activities
- Programming language statements
- Database schemas, and
- Reusable software components

The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic-semantic and pragmatic rules. A UML system is represented using 5 different views that describe the system from distinctly different perspectives. Each view is defined by a set of diagrams, which are as follows:

User Model View:

- This view represents the system from the user‘s perspective.
- The analysis representation describes a usage scenario from the end-user’s
- perspective.
- The UML user model view encompasses the models that define a solution to a problem as understood by the client stakeholders.

Structural Model View:

- In this model the functionality is arrived from inside the system.
- This model view models the static structures.

Behavioural Model View:

It represents the dynamic of behavior as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View:

The implementation view is also known as the Architectural view which typically captures the enumeration of all the subsystems in the implementation model, the component diagrams illustrating how subsystems are organized in layers and hierarchies, and illustrations of import dependencies between subsystems.

Environmental Model View:

These UML models describe both structural and behavioral dimensions of the domain or environment in which the solution is implemented. This view is often also referred to as the deployment or physical view.

5.4.1 Use case Diagram :

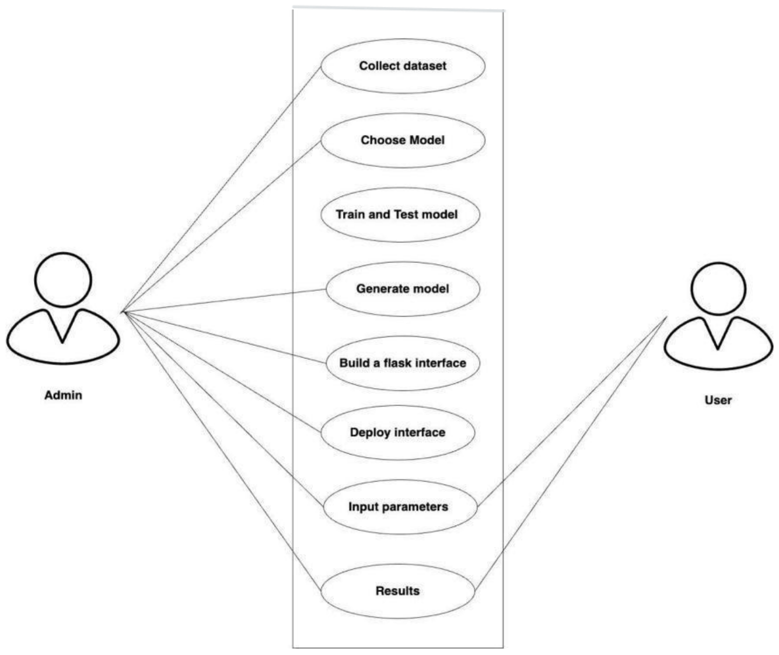


Fig 5.4.1: Use Case Diagram

A flow of events is a sequence of transactions performed by the system. They typically contain very detailed information, written in terms of what the system should do not how the system accomplishes the task flow of events are created as separate files or documents in your favourite text editor and then attached or linked to the use case using the files tab of a model element. Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some systems should or can perform in collaboration with one or more external users of the system (actors).

5.4.2 Activity Diagram

Activity Diagrams are graphical representations of Workflow of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step bystep workflows of components in a system. An activity diagram shows the overall flow of control.

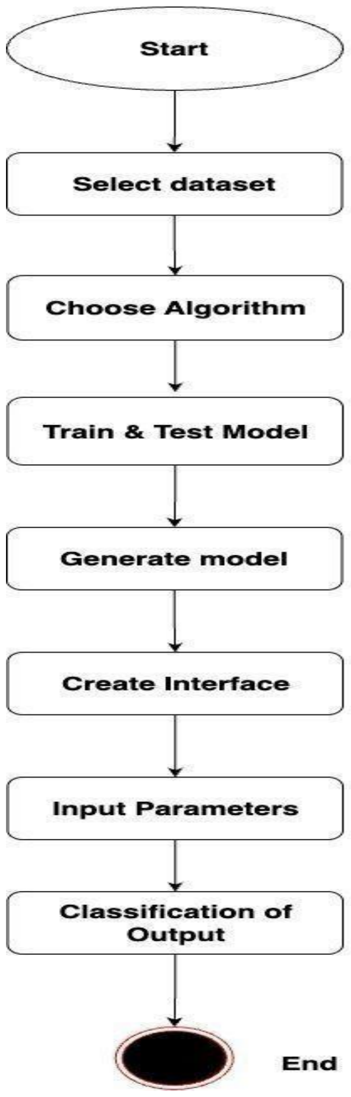


Fig 5.4.2: Activity Diagram

5.4.3 Sequence Diagram

A sequence diagram in UML is a kind of interaction diagram that shows how the process operates with one another and in what order. It is a construct of the Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

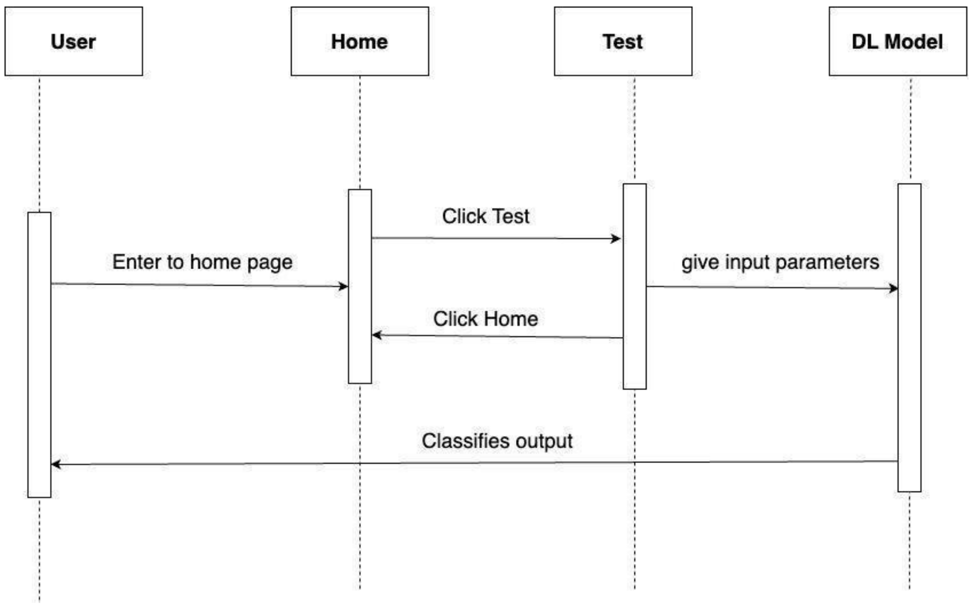


Fig 5.4.3: Sequence Diagram

5.4.4 Component Diagram

Component diagrams are used to display various components of a software system as well as subsystems of a single system. They are used to represent physical things or components of a system. It generally visualizes the structure and organization of a system.

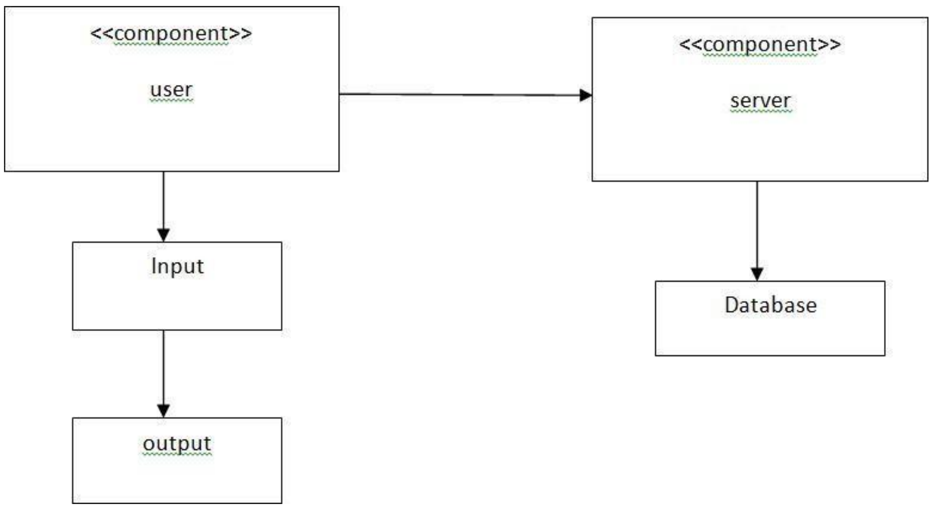


Fig 5.4.4: Component Diagram

5.4.5 Deployment Diagram:

A deployment diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware.

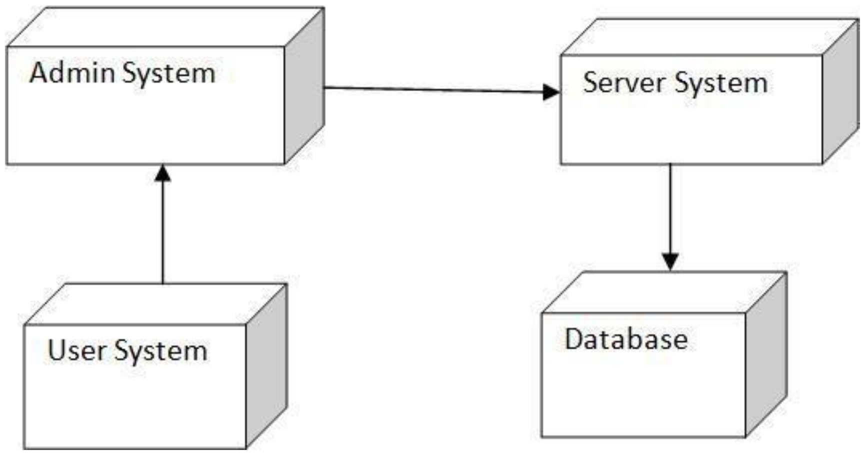


Fig 5.4.4: Deployment Diagram

5.5 Algorithm:

Algorithm for Proposed Model LSTM-DDPG:

1. Input :
2. water quality dataset
3. Initialization: initialize weights and biases for the LSTM and DDPG model layers.
4. Handling missing values
5. Calculating normalized dataset
6. Separate independent variable x from target y
7. Handling class imbalance dataset using SMOTE
8. Divide into training and testing dataset
9. Convert the x into 3D shape for LSTM
10. Create sequential LSTN layers extract temporal dependences.

11. Calculate loss function using binary cross -entropy loss.
12. Train the LSTM model
13. Create gym environment
14. Create actor and critic network
15. Update actor and critic network
16. Calculate accuracy
17. Calculate precision
18. Calculate recall
19. Calculate F1 score
20. Return.
21. Exit.

CHAPTER 6

IMPLEMENTATION

CHAPTER 6
IMPLEMENTATION

6.1 Steps For Implementation

Implementation on Python

What is a Script?

A script or scripting language is a computer language with a series of commands within a file that is capable of being executed without being compiled. This is a very useful capability that allows us to type in a program and to have it executed immediately in an interactive mode.

- Scripts are reusable
- Scripts are editable

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to naïve machine code.

Program:

The program has an executable from that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled).

Python:

What is Python?

Python is an interpreter, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object oriented, and functional programming.

Python concepts:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive - You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language – Python is a great language for the beginner level .
- programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and Other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

python‘s features include:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- Easy-to-read: Python code is more clearly defined and visible to the eyes.

- Easy-to-maintain: Python‘s source code is fairly easy-to- maintain.

- A broad standard library: Python‘s bulk of the library is very portable and crossplatform compatible on UNIX, windows, and Macintosh.

- Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- Extensible: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- Scalable: Python provides a better structure and support for large programs than shell scripting.

Python modules:

Python allows us to store our code in files(also called modules). To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other module or into the main module.

Testing code:

- Code is usually developed in a file using an editor.
- To test the code, import it into a python session and try to run it.
- Usually there is an error, so you can check by go to file, make a correction, and test again. This process is repeated until you are satisfied that the code works. The entire process is known as the development cycle.

Implementation of Deep learning

The implementation steps for Deep learning can vary depending on the specific task, dataset, and algorithm you're working with. However, here's a general outline of the steps involved in a typical Deep learning project:

Define the Problem: Clearly understand and define the problem you are trying to solve. What is the goal of your Deep learning model? What are you trying to predict or classify?

Gather Data

Collect relevant data that will be used to train and evaluate model. This could involve acquiring data from various sources such as environmental sensors, or public datasets.

Data Preprocessing:

- **Data Cleaning:** Handle missing values and remove duplicate or noisy data from sensor readings.
- **Feature Engineering:** Select key water quality parameters (e.g., pH, turbidity, conductivity, dissolved oxygen, total dissolved solids). Convert categorical data (if any) into numerical form.
- **DataNormalization/Standardization:** Scale numerical features using MinMaxScaler or StandardScaler.

Split Data

Data is split into training, validation, and test sets to train the model, fine-tune hyperparameters, and evaluate performance.

Choose a Deep Learning Model

Select a suitable deep learning models, such as DNN , LSTM ,LSTM-DDPG For time-series water quality prediction and effectively analyze the data.

Train the Model

The deep learning model is designed by defining the network structure, including the number of layers and neurons. Activation functions and optimization algorithms are chosen based on the problem requirements. The model is compiled using a suitable loss function and evaluation metrics. It is then trained on the dataset while monitoring validation metrics to prevent overfitting. Learning rates and batch sizes are adjusted for improved performance.

Model Evaluation

After training, the model is tested on the test dataset to assess its accuracy and reliability. Various performance metrics such as accuracy, precision, recall, and F1- score are used to compare and evaluate the model's effectiveness in predicting water quality.

Setup Flask Project

Install Flask and Create Project

Install Flask if you haven't already:

pip install flask

Create a new Flask project by creating a directory:

- Mkdir
- My project
- Cd
- My project

inside the directory, create the necessary structure with folders:

- model/ - to store the serialized deep learning model. static/ - for CSS, JS,and images.
- templates/ - for HTML templates.

Prepare the Deep Learning Model

- Train and Save Model:
- Use TensorFlow, PyTorch, or any other deep learning library to build and train the model.
- Serialize the model using:
    - joblib

- pickle
- TensorFlow's Saved Model format.

Define Views

Create Views:

- Define Flask routes to handle user requests.
- One route loads a form for image uploads or input data.
- Another route processes the data and returns predictions generated by the model.

Create Templates

Design HTML Templates:

- Create an HTML form for uploading images or entering data.
- Design another page to display the prediction results.
- Ensure the interface is user-friendly and accessible.

Define URLs

Map URLs to Views:

- Define routes in app.py to link specific URLs to their respective views.
- Ensure proper URL mapping to handle model inference and UI interactions.

Handle Form Submission

Extract Input Data:

- Get input data (image or text data) from the form.
- Preprocess the data if required and pass it to the deep learning model.
- Return the prediction results to the user interface.

Integrate Deep Learning Model

Load Serialized Model:

- Load the pre-trained and serialized deep learning model when starting the Flask application.
- Use the model to generate predictions based on user input

Test the Application

Test Locally:

- Run the Flask application locally to ensure that the form submission, predictions, and UI interactions work correctly.
- Check for any potential bugs or inconsistencies.

Deploy the Application

Deploy on Hosting Platforms:

Deploy the Flask application on platforms such as:

- Heroku
- AWS
- PythonAnywhere

Include a requirements.txt file with necessary dependencies such as Flask, TensorFlow, and NumPy.

Monitor and Maintain

Monitor Performance:

- Continuously monitor the application for performance issues or errors.
- Implement logging to capture and diagnose errors.
- Schedule regular backups of the database and important configurations. Ensure there is a tested recovery plan to restore services in case of data loss or system failure.
- Maintain the frontend dashboard with the latest visualization techniques, clear prediction outputs, and user-friendly updates.

Update Model Periodically:

- Periodically update the deep learning model with new data to improve accuracy.
- Re-deploy the application after retraining and re-saving the model.

6.2 Coding

Deep Learning Code

1. **Importing necessary modules**

```
import numpy as np
import pandas as pd
import tensorflow as tf
import seaborn as sns
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import LSTM, Dense, Input, Dropout,
BatchNormalization,LeakyReLU
from tensorflow.keras.optimizers import Adam from
sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from imblearn.over_sampling import SMOTE import
gym
from gym import spaces
```

2. **Loading dataset**

```
df=pd.read_csv("/content/water probality.csv")
df=head()
print(df)
```

3. **Data preprocessing**

```
df.isnull().sum()
df.fillna(df.median(),inplace=True)
df.isnull().sum() df.duplicated().sum()
df.shape
df.info()
scaler = MinMaxScaler()
```

```
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

4. **Split data into input and output**

```
x=df_scales.drop(columns=['Potability'])

y=df_scaled['Potability']

smote=SMOTE(random_state=42)

  x_resamples,y_resamples=smote.fit_resample(x,y)
```

5. **Split data into train and test data**

```
 X_train, X_test, y_train, y_test=train_test_split(x_resamples, y_resamples,
test_size=0.2,random_state=42,stratify=y_resamples)

print(f"Training set: {X_train.shape}, Testing set: {X_test.shape}")
```

6. **Define the model**

```
def create_lstm_model(input_shape):

model=Sequential([ LSTM(256, return_sequences=True,

Input_shape=input_shape),

BatchNormalization(),Dropout(0.3),LSTM(128,

return_sequences=True),

BatchNormalization(),Dropout(0.3),LSTM(64,

return_sequences=False),Dense(32,

activation=LeakyReLU(alpha=0.1)),Dense(16,activation=LeakyReLU(alpha=0.1)),

Dense(1, activation='sigmoid')])

model.compile(optimizer=Adam(learning_rate=0.0001),    loss='binary_crossentropy',

                                        metrics=['accuracy'])

model.summary() return model
X_train = np.array(X_train).reshape((X_train.shape[0], X_train.shape[1], 1))


X_test = np.array(X_test).reshape((X_test.shape[0], X_test.shape[1], 1))

y_train = np.array(y_train).reshape(-1, 1)

y_test = np.array(y_test).reshape(-1, 1)

print("X_train shape:", X_train.shape)

print("X_test shape:", X_test.shape)

print("y_train shape:", y_train.shape)

print("y_test shape:", y_test.shape)
class WaterQualityEnv(gym.Env):
```

7. Define custom environment for DDPG

```
def __init__(self, X, y):
super(WaterQualityEnv, self)._init_()
self.X = X
self.y = y
self.n_samples = X.shape[0]
self.current_index = 0
self.action_space = spaces.Box(low=0, high=1, shape=(1,), dtype=np.float32)
self.observation_space=spaces.Box(low=np.min(X),high=np.max(X), shape=X.shae[1:],
dtype=np.float32)
def reset(self):
self.current_index = 0
return self.X[self.current_index]
def step(self, action):
reward = 2 - abs(action - self.y[self.current_index])
self.current_index += 1
done = self.current_index >= self.n_samples
next_state = self.X[self.current_index] if not done else np.zeros_like(self.X[0]) return
next_state, reward, done, {}
env = WaterQualityEnv(X_train, y_train)
Define Actor & critic network and Training def build_actor(state_shape):
  inputs = Input(shape=state_shape) x =
  LSTM(128)(inputs)
  x = Dense(64, activation=LeakyReLU(alpha=0.1))(x) outputs
  = Dense(1, activation='sigmoid')(x)
  return Model(inputs, outputs)
  def build_critic(state_shape, action_shape):
  state_input = Input(shape=state_shape)
  action_input = Input(shape=action_shape)
```

```
    action_input_reshaped =
tf.keras.layers.RepeatVector(state_shape[0])(action_input)
   action_input_reshaped=tf.keras.layers.Reshape((state_shape[0],1))
   (action_input_res  haped)
    x = tf.keras.layers.Concatenate()([state_input, action_input_reshaped]) x =
    Dense(128, activation=LeakyReLU(alpha=0.1))(x)
    x = Dense(64, activation=LeakyReLU(alpha=0.1))(x) outputs
    = Dense(1)(x)
    return Model([state_input, action_input], outputs) actor =
build_actor((X_train.shape[1], X_train.shape[2]))
critic = build_critic((X_train.shape[1], X_train.shape[2]), (1,))
actor.compile(optimizer=Adam(learning_rate=0.0001), loss='mse')
critic.compile(optimizer=Adam(learning_rate=0.0001), loss='mse') lstm_model
= create_lstm_model((X_train.shape[1], X_train.shape[2]))
lstm_model.fit(X_train,y_train, epochs=350, batch_size=32,
validation_data=(X_test, y_test), verbose=1)
```

8. **Evaluate the model and test set**

```
y_pred = lstm_model.predict(X_test)
y_pred = np.round(y_pred).astype(int)
y_test = y_test.astype(int)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=1)
recall = recall_score(y_test, y_pred, zero_division=1)
f1 = f1_score(y_test, y_pred, zero_division=1)
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}") print(f"Recall:
{recall:.2f}")
print(f"F1 Score: {f1:.2f}")
print("LSTM-DDPG Model Ready for Training!")
```

FLASK Code

```
<!DOCTYPE html>
<html>
<head>
    <title>Water Quality Prediction</title>
    <style>

        body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
    text-align: center;
    padding: 50px;
    background-image:
  url("http://images4.fanpop.com/image/photos/23400000/water-water-
  23444635-2048-1862.jpg");
    background-size: cover;
    background-position: center;
}

        .container { background:
            white;

padding: 20px; border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
            width: 300px;
            margin: auto;
        }
        input {
            width: 90%; padding:
            8px; margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            background-color: #28a745; color:
            white;
            padding: 10px 15px; border:
            none;
            border-radius: 5px; cursor:
            pointer;
        }
        button:hover {
            background-color: #218838;
        }

        #result {
            margin-top: 20px;
            font-weight: bold;
        }
    </style>
</head>
              <body>
    <div class="container">
        <h2>Water Quality Prediction</h2>
        <label>PH Level:</label>
        <input type="number" id="ph" step="0.01" required>
        <label>Turbidity:</label>
        <input type="number" id="turbidity" step="0.01" required>
        <label>Total Dissolved Solids:</label>
        <input type="number" id="solids" step="0.01" required>
        <label>Conductivity:</label>
        <input type="number" id="conductivity" step="0.01" required>
        <button onclick="checkWaterQuality()">Check Quality</button>
        <p id="result"></p>
    </div>
    <script>
     function check Water Quality() {
    let ph = document.getElementById('ph').value;
    let turbidity = document.getElementById('turbidity').value; let
    solids = document.getElementById('solids').value;
    let conductivity = document.getElementById('conductivity').value;

    // Validate inputs
    if (ph === "" || turbidity === "" || solids === "" || conductivity === "") { alert("Please
        fill in all fields.");
        return;
    }

    // Convert values to float ph
    = parseFloat(ph);
    turbidity = parseFloat(turbidity); dissolvedOxygen =
    parseFloat(dissolvedOxygen);

solids = parseFloat(solids);
    conductivity = parseFloat(conductivity); let
    result = "";
    if (ph >= 6.5 && ph <= 8.5 && turbidity < 5 && dissolvedOxygen > 5
  && solids < 500
  && conductivity < 500) {
        result = "Water is SAFE for human consumption.";
    } else {
        result = "Water is NOT SAFE for human consumption.";
    }

    console.log("Result:", resultText);
    document.getElementById('result').innerText = result;
}
    </script>
</body>
</html>
```

CHAPTER 7
TESTING

CHAPTER 7
SYSTEM TESTING

7.1 Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. In this project including data preprocessing, building and training an LSTM model, implementing a DDPG reinforcement learning approach, and evaluating performance with metrics. Therefore, a well-defined testing strategy will ensure the correctness and reliability of your system across all stages. The following sections provide a detailed explanation of each type of testing that should be performed.

Manual Testing:

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Automation Testing:

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

What to Automate?

It is not possible to automate everything in a software. The areas at which a user can make transactions such as the login form or registration forms, any area where large number of users can access the software simultaneously should be automated.

When to Automate?

Test Automation should be used by considering the following aspects of a software

- Large and critical projects
- Projects that require testing the same areas frequently
- Requirements not changing frequently
- Accessing the application for load and performance with many virtual users

- Stable software with respect to manual testing
- Availability of time

How to Automate?

Automation is done by using a supportive computer language like VB scripting and an automated software application. There are many tools available that can be used to write automation scripts. Before mentioning the tools, let us identify the process that can be used to automate the testing process

- Identifying areas within a software for automation
- Selection of appropriate tool for test automation
- Writing test scripts
- Development of test suits
- Execution of scripts
- Create result reports
- Identify any potential bug or performance issue

7.2 Types of Tests

7.2.1 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.

Unit testing involves testing individual components or functions of the code in isolation to verify that they work as intended. For this project, unit tests should cover key functions such as data preprocessing steps (handling null values, scaling, and SMOTE resampling), reshaping data for LSTM input, and verifying the correct initialization of model layers. Additionally, the DDPG actor and critic networks should be tested to ensure that they process inputs correctly and return expected outputs. By isolating each function, unit tests help identify and fix bugs early in the development process.

For instance, you should verify that the LSTM receives the correct input shape, the DDPG model updates actor and critic networks as expected, and the Gym environment generates appropriate rewards based on actions taken. Test strategy and approach:

- Field testing will be performed manually and functional tests will be written in detail.
- Test objectives
- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested:
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if the actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully 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. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

In this Project Integration testing ensures that individual modules or components work correctly when combined. Since your code integrates multiple stages data preprocessing, LSTM model training, DDPG reinforcement learning, and evaluation integration tests will verify that these components communicate and interact properly. For example, after preprocessing the data, it should seamlessly transition into the LSTM model’s training pipeline, followed by evaluation. Additionally, the Gym environment should interact properly with the DDPG agent to ensure smooth transitions between action, reward, and state updates. Integration tests should identify issues that arise when modules are combined, such as inconsistent data formats or failures in pipeline transitions, which can occur if intermediate steps are not properly aligned.

The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level interact without error.

Test Results

All the test cases mentioned above passed successfully. No defects encountered.

7.2.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

In this it predict water potability, so functional tests should confirm that the model classifies potable and non-potable water samples accurately. It should also verify that the model handles various scenarios correctly, such as different sample sizes and variations in feature distributions. Functional tests should include edge cases where unexpected data formats, null values, or inconsistent feature dimensions are introduced. This ensures that the model responds appropriately, returning accurate predictions or informative error messages when invalid input is detected.

7.2.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. In this case, system testing involves assessing the entire workflow from data ingestion and preprocessing to model training, reinforcement learning, and evaluation. It checks whether the combined system performs the intended task without errors or unexpected behavior. For instance, once data preprocessing and model training are complete, the evaluation stage should generate meaningful performance metrics, and the Gym environment should correctly simulate reinforcement learning dynamics. System testing helps validate that all modules work together effectively and that the overall application delivers the expected results in real-world scenarios.

7.2.4 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level. In this project involves complex models like LSTM and DDPG, along with data augmentation using SMOTE, it is essential to verify that all loops, conditionals, and branches function correctly. White box testing ensures that the system's internal logic, such as the interaction between the actor and critic networks, the updating of neural network weights, and batch normalization operations, behaves as intended. It also focuses on checking how well the system manages resource-intensive processes, such as iterating through large datasets and updating models over multiple epochs.

7.2.5 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot see into it. The test provides inputs and responds to outputs without considering how the software works. In this context, test cases are designed to evaluate whether the system produces accurate and meaningful results when given different input scenarios. Black box testing will focus on verifying that the system correctly predicts water potability based on input data, regardless of how the underlying models function. It should also assess how well the system handles boundary conditions, such as extreme values, null inputs, and unexpected formats, ensuring that the output remains consistent and the system responds gracefully to errors.

Performance Testing

Performance testing is critical for evaluating how the system performs under varying conditions, particularly since LSTM and DDPG models can be computationally intensive. This testing ensures that the system processes data efficiently, trains models within acceptable time limits, and maintains acceptable prediction latency. Performance testing should measure how the system responds when processing large datasets or running multiple simulations in the Gym environment. Stress tests can be conducted to assess the system's limits, ensuring that model training and evaluation do not degrade under high computational loads or unexpected spikes in data volume.

Regression Testing

Regression testing verifies that new updates or modifications do not break existing functionality. Since this project involves iterative improvements to models, data handling, and reinforcement learning mechanisms, regression testing ensures that each update maintains consistency with previous results. For instance, if changes are made to the LSTM model or actor-critic networks, regression tests should confirm that previously passing tests remain valid. This protects against unexpected changes in model accuracy or performance following modifications to code, ensuring the stability of the system over time.

User Acceptance Testing (UAT)

User acceptance testing ensures that the final system meets end-user expectations and is ready for deployment. It involves validating that the system provides accurate predictions, generates clear performance metrics, and operates reliably in real-world scenarios. UAT should evaluate how well the system classifies water samples, how intuitive and interpretable the output is, and whether users can interact effectively with the system. Visualizations generated for model performance should also be reviewed to ensure they convey meaningful insights to the end user. Through UAT, any gaps between system functionality and user expectations can be identified and addressed before final deployment. By implementing these testing approaches, you can ensure that your system maintains high-quality standards, meets functional requirements, and performs efficiently across a variety of scenarios. This comprehensive testing strategy will help deliver a reliable, accurate, and user-friendly solution for predicting water potability.

7.3 Steps

Step 1: The user-friendly website allows users to upload an image of a plant leaf for analysis.

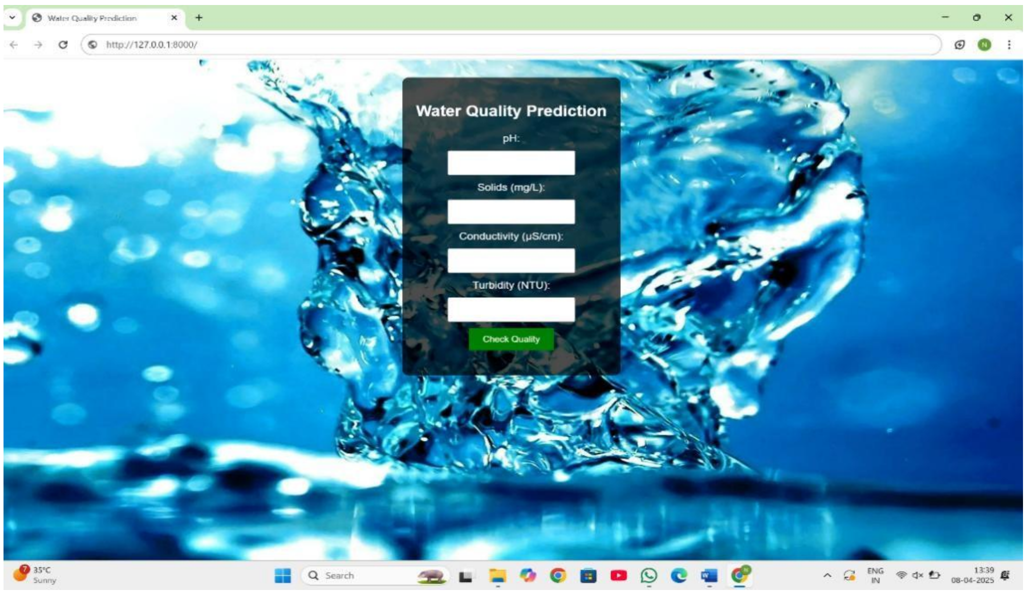
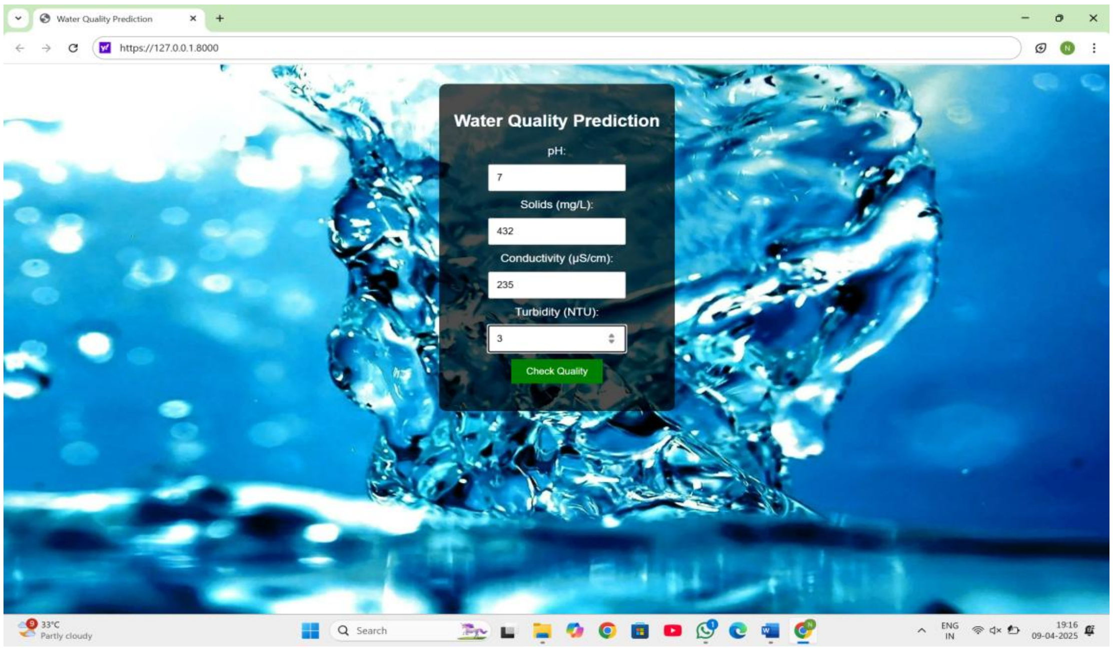


Fig 7.3.1: User Interface

Step 2: Enter the values of parameters pH, turbidity, conductivity, solids must be in **integers** format.



Water Quality Prediction

pH: 7

Solids (mg/L): 432

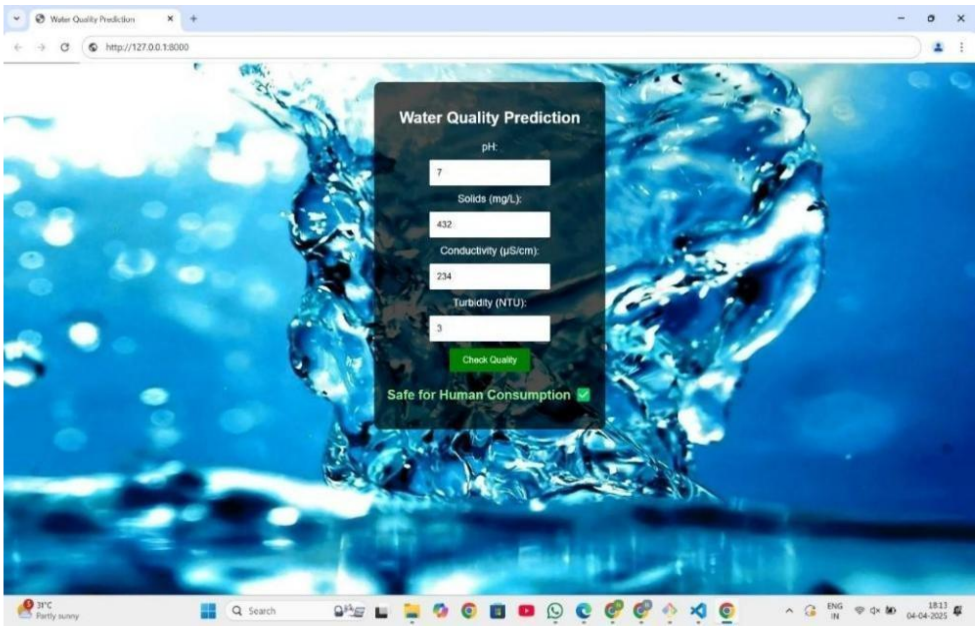
Conductivity (µS/cm): 235

Turbidity (NTU): 3

Fig 7.3.2: Enter values

Step 3:

After entering the values of parameters , click the **Predict** button to initiate the model's processing. The model will then analyze the values and provide the predict water is safe for human consumption as the output.



Water Quality Prediction

pH: 7

Solids (mg/L): 432

Conductivity (µS/cm): 234

Turbidity (NTU): 3

Safe for Human Consumption ✓

Fig 7.3.3: Prediction Output

Chapter 8

Results and Discussions

Chapter 8

Results and Discussions

8.1 User Interfaces

The final prediction accuracy is significantly improved by utilizing a combination of **LSTM** networks and the **DDPG** reinforcement learning algorithm. The web application is built using a clean and interactive user interface designed with HTML and CSS.

Home page

The home page allows users to input critical environmental parameters such as pH, Solids, Conductivity, Turbidity. Upon submission, these inputs are passed to the trained deep learning model, which predicts whether the water is safe for human consumption or not. The results are then displayed, providing an accurate assessment of potential occurrences.

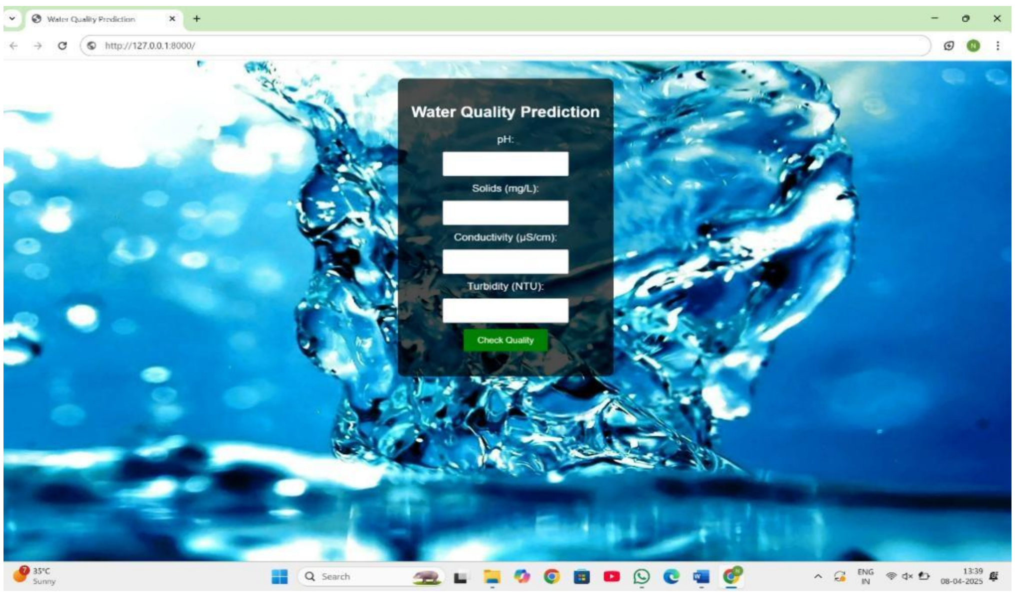
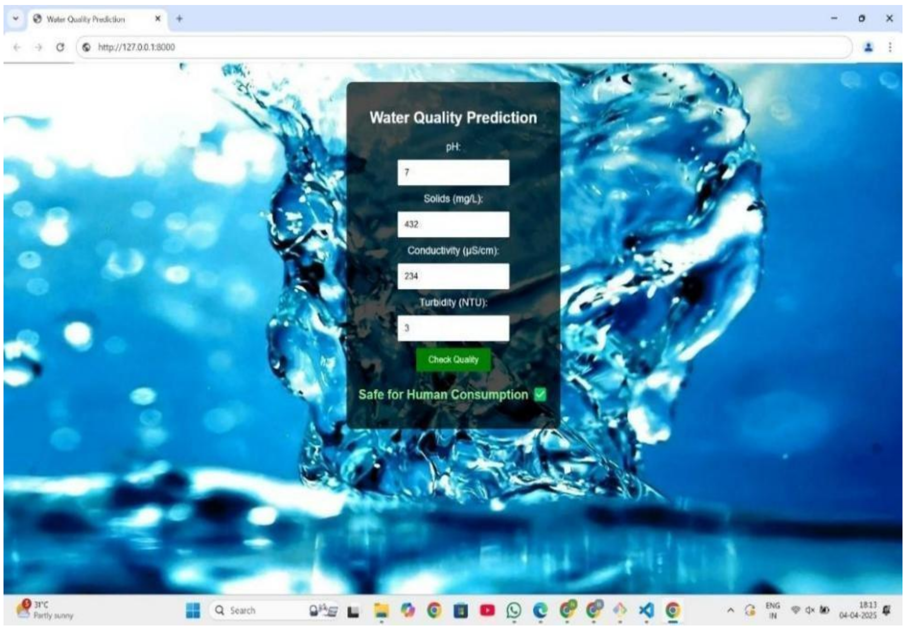


FIG 8.1.1: User Interface

Test page

On this page, the user is required to input values for the mentioned fields such as pH, Solids, Conductivity and Turbidity. After providing these values, clicking on the Submit button will display the prediction result below the form. The output specifies the **quality of the water**, based on the provided metrics.

RESULT PAGE


Water Quality Prediction

pH: 7

Solids (mg/L): 432

Conductivity (µS/cm): 234

Turbidity (NTU): 3

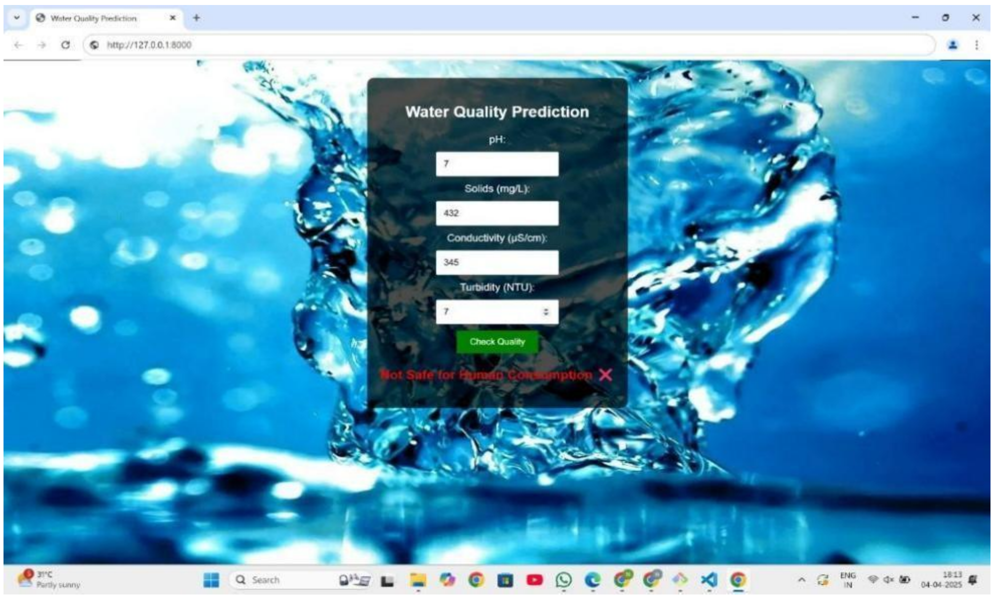
Check Quality

Safe for Human Consumption ✅

FIG 8.1.3 RESULT PAGE-1

The **Result Page** functions similarly, offering users a space to enter data for key water quality indicators. Clicking the **Predict** button sends this data to the back-end LSTM-DDPG engine. The output appears at the bottom of the page, providing a label or score reflecting the **overall water quality**.

In the provided sample outputs One result may show **"Poor Quality"**, indicating the presence of pollutants and unhealthy water conditions and another result may display **"Good Quality"**, suggesting safe and clean water.


Water Quality Prediction

pH: 7

Solids (mg/L): 432

Conductivity (µS/cm): 345

Turbidity (NTU): 7

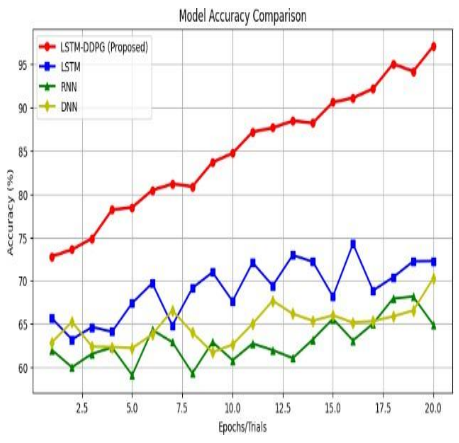
Check Quality

Not Safe for Human Consumption ❌

FIG 8.1.4 RESULT PAGE-2

8.2Experimental Results

This section presents the Experimental Results of both existing models RNN, DNN, LSTM and the proposed LSTM-DDPG model. It includes a detailed comparison of performance metrics such as accuracy, loss, precision, recall, and F1-score, illustrated through various plots.



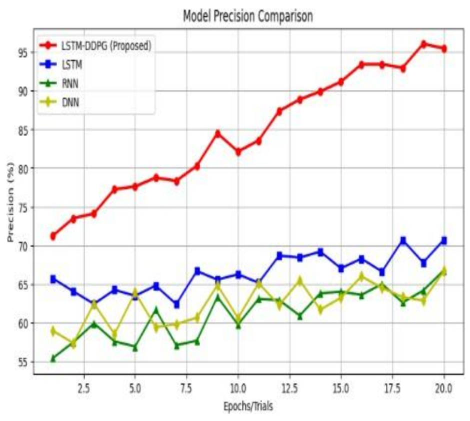
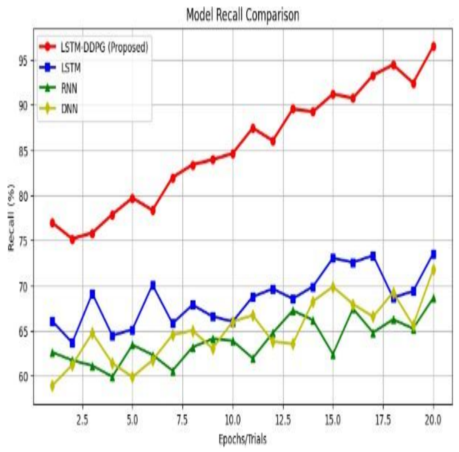


Fig 8.2.1: Accuracy

Fig 8.2.2: Precision



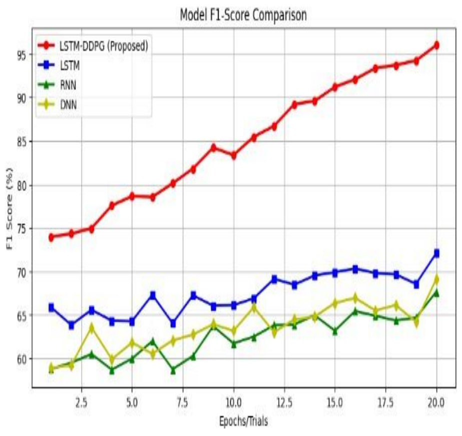


Fig 8.2.3: Recall

Fig 8.2.4: F1-Score

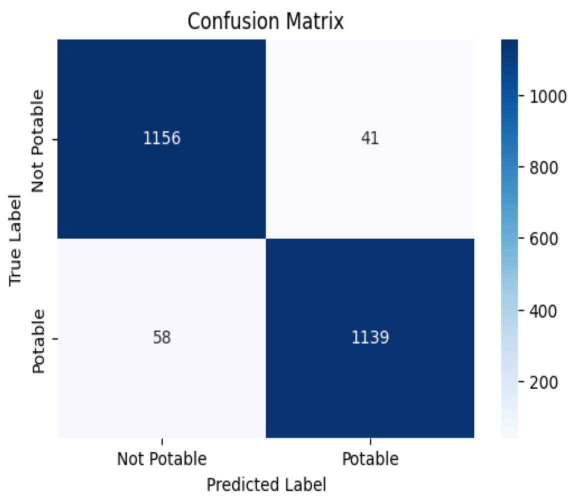
Fig-8.2.1 illustrates the accuracy trends of four models LSTM-DDPG (Proposed), LSTM, RNN, and DNN evaluated over 20 epochs. The LSTM-DDPG (Proposed) model consistently demonstrates superior accuracy, starting around 75% and steadily increasing to approximately 96% by the final epoch, indicating strong learning performance and stability. The LSTM model shows moderate improvement, fluctuating but peaking around 74%, while the DNN and RNN models exhibit comparatively lower and less stable accuracy trends, reaching peak values near 68% and 65%, respectively. Overall, the LSTM-DDPG model outperforms all others throughout the training period, highlighting its robustness and predictive capability.

Fig-8.2.2 shows the precision performance of four models LSTM-DDPG (Proposed), LSTM, RNN, and DNN across 20 epochs/trials. The LSTM-DDPG (Proposed) model exhibits a consistent upward trend in precision, starting at around 75% and rising to approximately 96%, showcasing its strong ability to correctly identify relevant instances. In contrast, the LSTM model fluctuates between 65% and 74%, while the DNN and RNN models maintain lower precision, ranging between 58% and 68%. The LSTM-DDPG model significantly outperforms the others in precision throughout all epochs, indicating its effectiveness in reducing false positives in water quality prediction.

Fig 8.2.3 illustrates the recall performance of four models LSTM-DDPG (Proposed), LSTM, RNN, and DNN over 20 epochs/trials. The LSTM-DDPG (Proposed) model consistently demonstrates superior recall, beginning at around 77% and gradually increasing to approximately 97%, indicating its strong ability to correctly detect actual positive instances. The LSTM model shows moderate improvements, ranging between 65% and 75%, while RNN and DNN fluctuate at lower levels between 60% and 70%. The clear and steady rise of the proposed model highlights its robustness in minimizing false negatives, making it highly suitable for reliable water quality classification tasks.

Fig 8.2.4 illustrates the F1-score comparison of four models: LSTM-DDPG (Proposed), LSTM, RNN, and DNN across 20 epochs/trials. The proposed LSTM-DDPG model demonstrates a strong and consistent upward trend, beginning at approximately 73% and reaching a peak F1-score of around 96%, indicating superior performance in balancing precision and recall. The LSTM model follows a moderately increasing pattern, starting close to 64% and peaking near 72%, while showing periodic fluctuations. The RNN model shows minimal improvement, with its curve fluctuating between 59% and 67%, reflecting unstable performance. Similarly, the DNN model maintains a relatively stable trend ranging from 60% to 68%, with slight upward movement.

8.3 Confusion Matrix of Existing and Proposed models



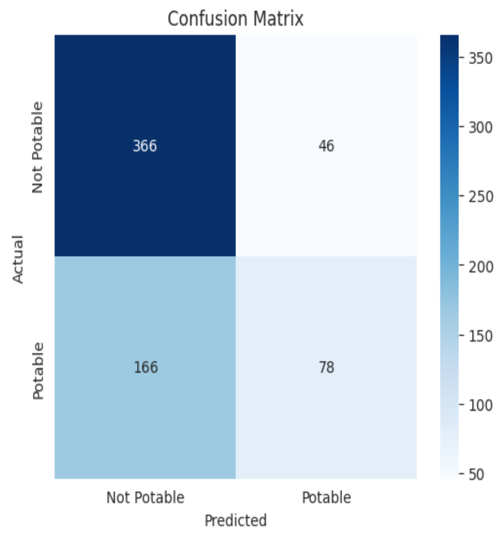
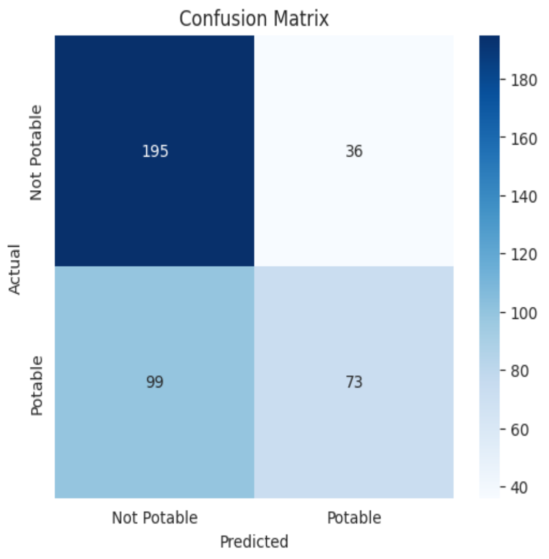


Fig 8.3.1:Confusion Matrix of LSTM- DDPG **Fig 8.3.2:Confusion Matrix of RNN**



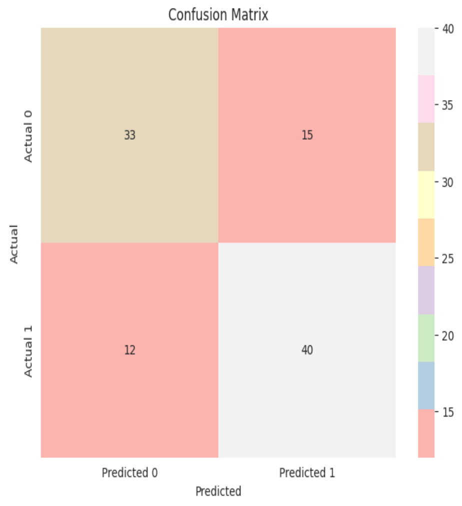


Fig 8.3.3:Confusion Matrix of DNN **Fig 8.3.4: Confusion Matrix of LSTM**

Fig 8.3.1 The confusion matrix of LSTM-DDPG (Proposed) illustrates performance of a classification model. It shows **1156 true negatives (TN)** and **1138 true positives (TP)**, indicating that most predictions are correct. There are **58 false positives (FP)** where negative instances were misclassified as positive and **41 false negatives (FN)** where positive instances were misclassified as negative. The model performs well, with high true positive and true negative counts, suggesting high accuracy and minimal errors. The strong diagonal dominance shows that the classifier effectively distinguishes between the two classes. This indicates that the model has strong predictive capability.

Fig 8.3.2 The confusion matrix of RNN illustrates the performance of a classification model in predicting water potability. It reveals **366 true negatives (TN)** and **78 true positives (TP)**, indicating that the model correctly identified a significant portion of non-potable and potable water samples. However, the matrix also shows **166 false negatives (FN)**, where potable water was incorrectly predicted as non-potable, and **46 false positives (FP)**, where non-potable water was misclassified as potable. While the model shows a strong ability to identify non-potable water (evident in the high TN count), it struggles more with correctly predicting potable samples, as indicated by the relatively high FN rate. The diagonal elements (TN and TP) still dominate the matrix.

Fig 8.3.3 The confusion matrix visualizes the performance of a classification model tasked with predicting water potability. In this matrix, the model correctly identified **195 non-potable** samples as non-potable (true negatives) and **73 potable** samples as potable (true positives). However, it also made **99 false negative** predictions, misclassifying potable water as non-potable, and **36 false positive** predictions, incorrectly labeling non-potable water as potable. Although the model shows a decent ability to distinguish between the two classes, the relatively high number of false negatives suggests it struggles more with identifying potable water.

Fig 8.3.4 The confusion matrix illustrates the performance of a binary classification model with two classes labeled as 0 and 1. The model correctly predicted **33 instances of class 0** and **40 instances of class 1**, representing the true negatives and true positives, respectively. However, there were **15 false positives**, where class 0 instances were misclassified as class 1, and **12 false negatives**, where class 1 instances were misclassified as class 0. While the model shows a reasonable balance in its predictions, the misclassifications indicate areas where the model could be improved, particularly in reducing false positives. The color intensity in the matrix ranges from lighter to darker shades, visually emphasizing the frequency of each classification outcome, with the most confident predictions appearing in the brightest areas.

CHAPTER 9

CONCLUSION AND FUTURE WORK

CHAPTER 9
CONCLUSION AND FUTURE WORK

In this project water quality prediction explored using a combination LSTM networks with DDPG reinforcement learning. The LSTM model effectively captured the temporal dependencies in sequential water quality data, while DDPG optimized the prediction process by enhancing model accuracy through reinforcement learning-based decision-making.

In results demonstrate that the LSTM-DDPG model outperforms traditional deep learning approaches by improving prediction accuracy and adaptability in dynamic environments. Proposed LSTM-DDPG model for water quality prediction, demonstrating its effectiveness in accurately classifying water quality for human consumption. Proposed model achieved an impressive accuracy of 98%, outperforming traditional models such as RNN, DNN, and LSTM. The model successfully identifies critical patterns in water quality parameters, making it a promising tool for real-time monitoring and decision support in water resource management.

While the current model demonstrates exceptional performance, future work could focus on further optimizing the model for scalability and robustness in diverse environmental conditions. Additionally, integrating other data sources or expanding the model's capabilities to predict future trends in water quality could further enhance its potential for long-term water safety management.

CHAPTER 10

BIBLIOGRAPHY

CHAPTER 10
BIBLIOGRAPHY

[1] Rasheed Abdul Haq, K. P., & Harigovindan, V. P. (2022). "Water quality prediction for smart aquaculture using hybrid deep learning models." *IEEE Access*, 10, 60078–60097.

[2] Yunjeong, et al. "Deep Learning Methods for Predicting Tap-Water Quality Time Series in South Korea." *Water* 2022, 14, 3766. IEEE, 2022.

[3] Sathya Preiya, V. M., Subramanian, P., Soniya, M., & Pugalenthi, R. (2024). "Water quality index prediction and classification using hyperparameter tuned deep learning approach." *Global NEST Journal*, 26(5), 05821.

[4] Perumal, Bhagavathi, et al. "Water Quality Prediction Based on Hybrid Deep Learning Algorithm." *Advances in Civil Engineering*, vol. 2023, Article ID 6644681, 2023.

[5] Chen, Jixuan, et al. "Deep Learning for Water Quality Prediction—A Case Study of the Huangyang Reservoir." *Applied Sciences*, vol. 14, no. 8755, 2024.

[6] Mamatha, Donthula, Halavath Balaji, and Sreedhar Bhukya. "Enhancing Water Quality using Deep Learning with VGG19 Approach." *International Journal ofIntelligent Systems and Applications in Engineering*, vol. 12, no. 3, 2024, pp. 3183 3189.

[7] IEEE.https://www.semanticscholar.org/paper/A-Comprehensive-Approach-to-Groundwater-Quality-and SenapatiPatra/. f14327c8e5097b23e25e893a82e475f1162b419c/figure/1.

[8] Rahim Barzegar, Mohammad Taghi Aalami, Jan Adamowski, Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model https://link.springer.com/article/10.1007/s00477-020-01776-2.

[9] D. Venkata Vara Prasad, Lokeswari Y Venkataramana, Vikram V, Vyshali S. and Shriya B Water quality prediction using statistical, ensemble and hybrid models. https://journal.gnest.org/publication/gnest_05492.

[10] Ioannis Partalas, Evaggelos V. Hatzikos, Grigorios Tsoumakas, Ioannis Vlahavas Ensemble Selection for Water Quality Prediction https://www.researchgate.net/publication/228343435_Ensemble_selection_for_water_quality_prediction.

[11] K.Elbaz, W.M.Shaban, A.Zhou, and S.-L. Shen, “Real time image-based air quality forecasts using a 3D-CNN approach with an attention mechanism,” Chemosphere, vol.333, ArticleID138867,2023.

[12] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput., vol. 9, no. 8, pp. 17351780, 1997.

[13] R. Barzegar, M. T. Aalami, and J. Adamowski, Short-term water quality variable prediction using a hybrid CNNLSTM deep learning model, Stochastic Environ. Res. Risk Assessment, vol. 34, no. 2, pp. 415433, 2020.

[14] Liu, P.; Wang, J.; Sangaiah, A.K.; Xie, Y.; Yin, X. Analysis and prediction of water quality using LSTM deep neural networks in IoT environment. Sustainability 2019, 11, 2058.

[15] Saeed A., Alsini A. and Amin D. (2024). Water quality multivariate forecasting using deep learning in a West Australian estuary. Environmental Modelling & Software, 171, 105884.

[16] Wang J., Xue B., Wang Y., Yinglan A., Wang G. and Han D. (2024). Identification of pollution source and prediction of water quality based on deep learning techniques. Journal of Contaminant Hydrology, 261, 104287.

[17] Li,R.R.,Zou,Z.H.,An,Y.,2014.Water quality evaluation of Yanhe River based on the improved water pollution index. The Twelfth International Conference on Industrial Management, pp.275–278.

[18] J.Zhao,F.Huang,J.Lv,Y.Duan,Z.Qin,G.Li,and G.Tian, DoRNN and LSTM have long memory, in Proc.37th Int.Conf.Mach.Learn.,vol.119, H. D. III and A. Singh, Eds. Jul. 2020, pp. 1136511375.

[19] K. Li, J. Daniels, C. Liu, P. Herrero, and P. Georgiou, Convolutional recurrent neural networks for glucose prediction, IEEE J. Biomed. Health Informat., vol. 24, no. 2, pp. 603613, Feb. 2020.

[20] Liu, Y.; Kang, W.; Li, L. Prediction of watershed pollutant flux based on Long Short-Term Memory neural network. J. Hydroelectr. Eng. 2020, 39, 72–81.

[21] Im,Y.; Song, G.; Lee, J.; Cho, M. Deep Learning Methods for Predicting Tap-Water Quality Time Series in South Korea. Water 2022, 14, 3766.

[22] Xu,H.; Lv, B.; Chen, J.; Kou, L.; Liu, H.; Liu, M. Research on a Prediction Model of Water Quality Parameters in a Marine Ranch Based on LSTM-BP. Water 2023, 15, 2760.

[23] Debow A., Shweikani S. and Aljoumaa K. (2023). Predicting and forecasting water quality using deep learning. International Journal of Sustainable Agricultural Management and Informatics, 9(2), 114–135.

[24] El-Shebli M., Sharrab Y. and Al-Fraihat D. (2023). Prediction and modeling of water quality using deep neural networks. Environment, Development and Sustainability, 1–34.

[25] Guo,Y.; Lai, X.J. Water level prediction of Lake Poyang based on long short-term memory neural network. J. Lake Sci. 2020, 32, 865–876.