1. INTRODUCTION

Flood is the most common disaster, causing losses of human life, infrastructure, damage to property, agriculture, and livestock. The process of forecasting and estimating the occurrence, intensity, and potential impact of flooding events in a particular area is known as flood forecasting. The aim of flood prediction is to provide prior information to communities, authorities, and individuals, allowing them to take preventive measures and mitigate the potential damage caused by floods. To predict the flood, a machine learning model is needed to analyze complex datasets, identify patterns, and improve prediction accuracy. Machine learning offers a wide range of approaches for prediction. Machine learning methods are utilized to predict floods by finding patterns in a set of data . Supervised, unsupervised, and reinforcement learning are the three categories into which machine learning approaches fall. In a supervised learning algorithm, a dataset is given as input to the algorithm, which is then processed and optimized to meet a set of specific outputs Unsupervised learning algorithm to analyze and cluster

Unlabeled dataset and reinforcement learning algorithm, self-trained on rewards and punishment mechanisms. The supervised learning algorithms are Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, Naive Bayes, , and so on . On the basis of the above study, it is observed that the supervised learning methods are showing the best results for prediction, which motivates us to choose the same for this article. In view of past precipitation information, the best of the two methodologies is picked for expectation dependable deep learning and ML model for a real- time flood detection system. It uses convolutional neural networks, random forests, and naive Bayes to identify water levels and evaluate floods that may have humanitarian implications before they happen . There are many advantages to the machine learning techniques for flood forecasting that are suggested in . Then, it develops an intuitive web interface system using an Intelligent Hydro Informatics Integration Platform to enhance online forecasting and flood risk management. This is achieved by utilizing machine learning, visualization, and system development methodologies. The most promising short- and long-term flood prediction methods are presented in . An investigation is also given to the significant advancements in improving the caliber of flood prediction models.

The researchers found that the best strategies for enhancing ML techniques were data decomposition, hybridization, ensemble modeling, and optimization. A useful flood modeling framework for simulations was implemented. With the use of a hybrid hydraulic model and algorithms for learning, the structure offers a novel, quick, effective, and expandable method for determining the level of floods. Thus, two machine learning models were used. presented Random Forest (RF) as a competitive substitute for Support Vector Machine (SVM) that often beats SVM in flood prediction models. developed an expert prediction model by utilizing the self-adaptive Evolutionary Extreme Learning (ELM) and a non-tuned machine learning method. Water level prediction is the main use of the SaE-primary ELM. Creating such models for water level prediction and monitoring is a crucial optimization issue in water resources management and flood.

prediction examined the effectiveness of random forest (RF), artificial neural network (ANN), and support vector machine (SVM) in general applications to floods and discovered that RF provided the greatest results. The support vector machine was utilized to forecast floods.

objectiv model performs better than the benchmark models in the absence of cutting-edge flood monitoring technology, it still requires a lot of work. A prediction model was developed using rainfall data to forecast the frequency of floods brought on by precipitation. Based on the range of rainfall in specific locations, the model predicts if a "flood" will happen. Using information on rainfall from districts in India, the forecast model will run. The dataset was trained using a variety of methods, including Multilayer perceptron, support vector machine, K-nearest neighbor, and linear regression. The MLP algorithm performed with a precision of 97.30%. Based mostly on upstream stage observation, ML models may forecast flood stages at a major gauge station . The case study for this analysis is the lower Parma River in Italy, and a 9-hour forecast horizon was used. Three machine learning algorithms were compared and processing speed: support vector regression (SVR), multi-layer perceptron (MLP).

To improve performance of flood prediction using supervised learning models. The article's contribution includes implementing various supervised machine learning models comparing the performance of logic regression with Random Forest, Extra Tree classifier, LGBM classifier, and CatBoost classifier.

2. SYSTEM ANALYSIS

2.1 Existing System

The existing flood prediction systems mainly rely on traditional forecasting methods such as hydrological models, rainfall-runoff models, and numerical weather prediction models. These systems are based on empirical data and predefined threshold values, which help in predicting floods based on historical rainfall and water levels. However, these models have limitations such as their dependency on accurate data, which is often not available in real time, and their inability to capture the complex non-linear relationships between weather patterns and flood events.Additionally, many existing systems use machine learning techniques, but these models are often limited in scope or focus on a single algorithm. Some of the more common machine learning techniques used include:

Disadvantages of the Existing System:

- Existing systems rely on outdated data and predefined thresholds, limiting accuracy.
- They struggle with handling large and complex datasets efficiently.
- Traditional models, like SVM and ANN, are prone to overfitting and lack generalization.
- There is limited real-time data integration, affecting prediction timeliness.
- The systems often lack user-friendly interfaces, making them difficult for non-experts to use.

Lack of Automation: Without automation, the process of updating models and making predictions is time-consuming and less scalable, limiting its usefulness for large datasets or rapid decision-making.

Existing system algorithms:

- Support Vector Machine (SVM)
- Artificial Neural Networks (ANN)
- Naive Bayes

2.2 Proposed System

The proposed system aims to overcome the limitations of existing flood prediction systems by utilizing a combination of advanced machine learning algorithms and real-time data integration. The focus is on improving prediction accuracy, scalability, and real-time application. The key features of the proposed system. . However, these models have limitations such as their dependency on accurate data, which is often not available in real time, and their inability to capture the complex non-linear relationships between weather patterns and flood events.Additionally, many existing systems use machine learning techniques, but these models are often limited in scope or focus on a single algorithm. Some of the more common machine learning techniques used include:

Advantages of the Proposed System:

- The proposed system integrates real-time data, providing more accurate and timely flood predictions.
- It uses advanced machine learning algorithms, improving prediction accuracy and handling complex datasets.
- Hybrid and ensemble methods are employed, enhancing model robustness and generalization.
- The system is scalable, allowing it to handle large datasets and adapt to evolving flood patterns.
- A user-friendly interface ensures easy access for both authorities and the general public.
- It uses advanced machine learning algorithms, improving prediction accuracy and handling complex datasets.
- Hybrid and ensemble methods are employed, enhancing model robustness and generalization.
- The system is scalable, allowing it to handle large datasets and adapt to evolving flood patterns.
- A user-friendly interface ensures easy access for both authorities and the general public.

3. SYSTEM REQUIREMENTS

- **REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

- **REQUIREMENT SPECIFICATION**

Functional Requirements

Graphical User Interface With The User.

3.1 SOFTWARE REQUIREMENTS

For developing the application the following are the Software Requirements:

- Python
- Django

Operating Systems supported

- Windows 10 64 bit OS

Technologies and Languages used to Develop

- Python

Debugger and Emulator

- Any Browser (Particularly Chrome)

3.2 HARDWARE REQUIREMENTS

For developing the application the following are the Hardware Requirements:

- Processor: Intel i5
- RAM: 8 GB
- Space on Hard Disk: minimum 500 GB

4. SYSTEM STUDY

4.1 FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

4.1.1 ECONOMICAL FEASIBILITY

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

4.1.2 TECHNICAL FEASIBILITY

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

4.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. also able to make some constructive criticism, which is welcomed, as he is the final user of the system

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. also able to make some constructive criticism, which is welcomed, as he is the final user of the system

Types of Feasibility Study

Feasibility studies can be categorized into five main types: technical feasibility, financial feasibility, market feasibility (also known as market fit), operational feasibility, and legal feasibility. A thorough feasibility study typically evaluates all five areas.

Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team can convert the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building currently, this project is not technically feasible.

Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent and enhances project credibility helping decision-makers determine the positive economic benefits to the organization the proposed project will provide.

Legal Feasibility

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine how a satisfies the requirements identified in the requirements analysis phase of system development.

Scheduling Feasibility

This assessment is the most important for project success; a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete. When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

- Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- Internal Corporate Constraints: Financial, Marketing, Export, etc.
- External Constraints: Logistics, Environment, Laws, and Regulations, etc.

5. SOFTWARE ENVIRONMENT

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a and comprehensive standard library.

5.1 INTERACTIVE MODE PROGRAMMING:

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

5.2 SCRIPT MODE PROGRAMMING:

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py      # This is to make file executable

$./test.py
```

This produces the following result −

```
Hello, Python!
```

5.3 PYTHON IDENTIFIERS:

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

5.4 RESERVED WORDS:

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambdayield	

5.5 LINES AND INDENTATION:

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:

print       "True"

else:

print "False"
```

However, the following block generates an error −

```
if True:

print   "Answer"

print       "True"

else:

print   "Answer"

print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import sys

try:

# open file stream

file = open(file_name, "w")

except IOError:

print "There was an error writing to", file_name

sys.exit()

print "Enter '", file_finish,

print "' When finished"

while file_text != file_finish:

file_text = raw_input("Enter text: ")

if file_text == file_finish:

 # close the file

file.close

break

file.write(file_text)

file.write("\n")

file.close()

file_name = raw_input("Enter filename: ")

if len(file_name) == 0:

print "Next time please enter something"

sys.exit()
```

try:

```
file = open(file_name, "r")

except IOError:
```

print "There was an error reading file"

sys.exit()

file_text = file.read()

file.close()

print file_text

5.6 MULTI-LINE STATEMENTS:

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
item_two        +        \
item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence   =   "This   is   a   sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.

# This is a comment, too.

# This is a comment, too.

# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''

This is a multiline

comment.

'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :

suite

elif expression :

suite

else :

suite
```

5.7 COMMAND LINE ARGUMENTS:

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h

usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d     : debug output from parser (also PYTHONDEBUG=x)

-E     : ignore environment variables (such as PYTHONPATH)

-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];
```

```
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0];

print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics

tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara

dict['Age']:  7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:

Traceback (most recent call last):

 File "test.py", line 4, in <module>

print  "dict['Alice']:  ",  dict['Alice'];

KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python

dict  =  {'Name':  'Zara',  'Age':  7,  'Class':  'First'}

dict['Age']    =    8;    #    update    existing    entry

dict['School'] = "DPS School"; # Add new entry
```

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

When the above code is executed, it produces the following result −

dict['Age']: 8

dict['School']: DPS School

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'} del

dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

dict['Age']:

Traceback (most recent call last):

File "test.py", line 8, in <module>

print "dict['Age']: ", dict['Age'];

TypeError: 'type' object is unsubscriptable

Note − del() method is discussed in subsequent section.

5.8 PROPERTIES OF DICTIONARY KEYS:

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):

File "test.py", line 3, in <module>

dict = {['Name']: 'Zara', 'Age': 7};

TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

# Following action is not valid for tuples

# tup1[0] = 100;

# So let's create a new tuple as follows

tup3 = tup1 + tup2;

print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong withputting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);

print tup;

del tup;

print "After deleting tup : ";

print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)

After deleting tup :

Traceback (most recent call last):

 File "test.py", line 9, in <module>

 print tup;
```

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

5.1.9 DJANGO WORKING:

Django follows the MVT design pattern (Model View Template).

- Model - The data you want to present, usually data from a database.
- View - A request handler that returns the relevant template and content - based on the request from the user.
- Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

➢ Model

The model provides data from the database.In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it easier to work with databases.The most common way to extract data from a database is SQL. One problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.
The models are usually located in a file called models.py.

➢ View

A view is a function or method that takes http requests as arguments, imports the relevant model(s), and finds out what data to send to the template, and returns the final result.
The views are usually located in a file called views.py.

➢ Template

A template is a file where you describe how the result should be represented.Templates are often .html files, with HTML code describing the layout of a web page, but it can also be in other file formats to present other results, but we will concentrate on .html files.
Django uses standard HTML to describe the layout, but uses Django tags to add logic:

```
<h1>My Homepage</h1>
<p>My name is {{ firstname }}.</p>
```

The templates of an application is located in a folder named templates.

➢ URLs

Django also provides a way to navigate around the different pages in a website.When a user requests a URL, Django decides which *view* it will send it to.This is done in a file called urls.py.Django is a high- level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

➢ COMPLETE

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive

➢ VERSATILE

Django can be (and has been) used to build almost any type of website from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, and XML).
Internally, while it provides choices for almost any functionality you might want (e.g., several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

➢ DJANGO HISTORY

Django was invented by Lawrence Journal-World in 2003, to meet the short deadlines in the newspaper and at the same time meeting the demands of experienced web developers.
Initial release to the public was in July 2005.
Latest version of Django is 4.0.3 (March 2022).

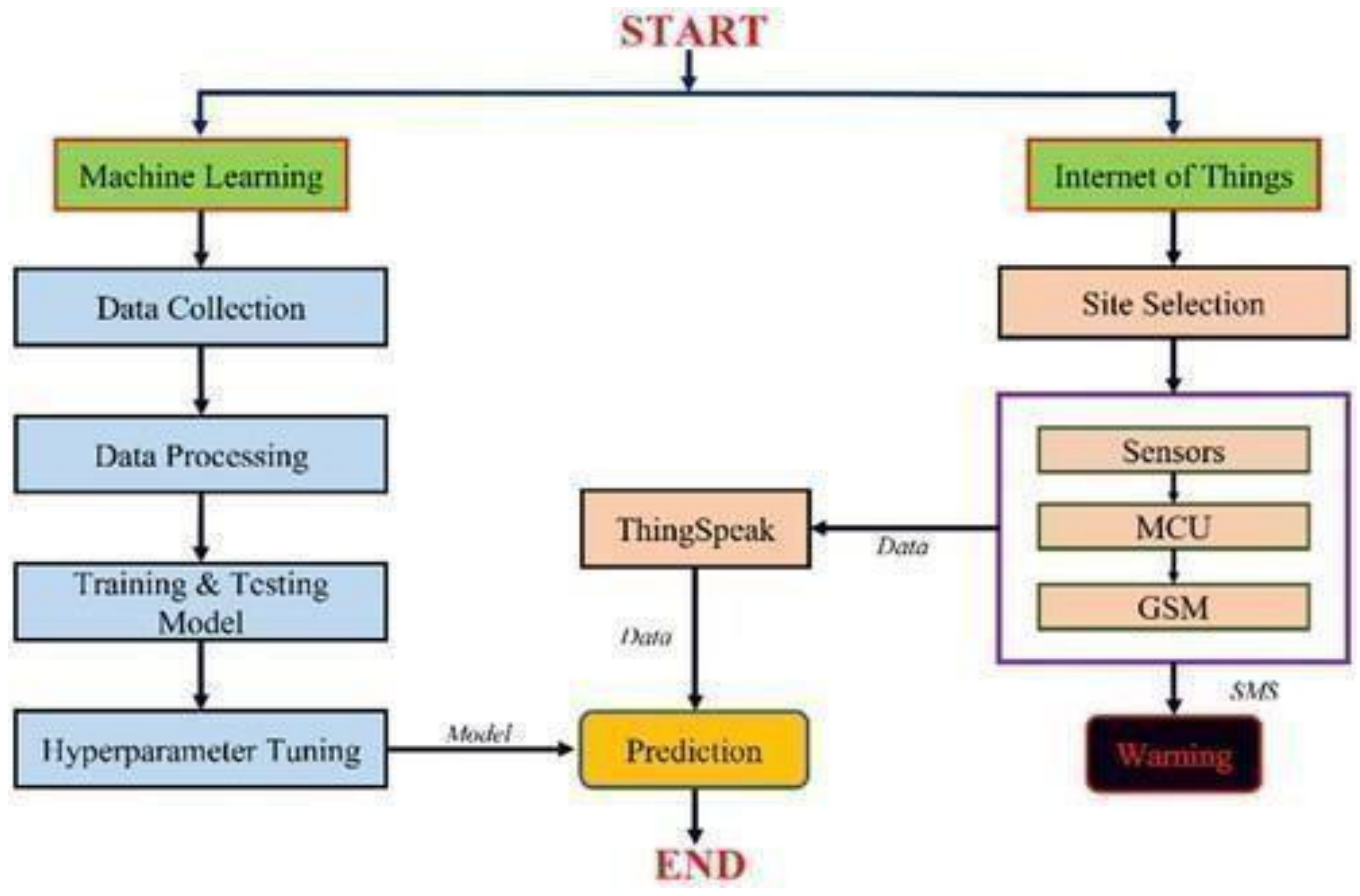


FIG :DJANGO DEPLOYMENT

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

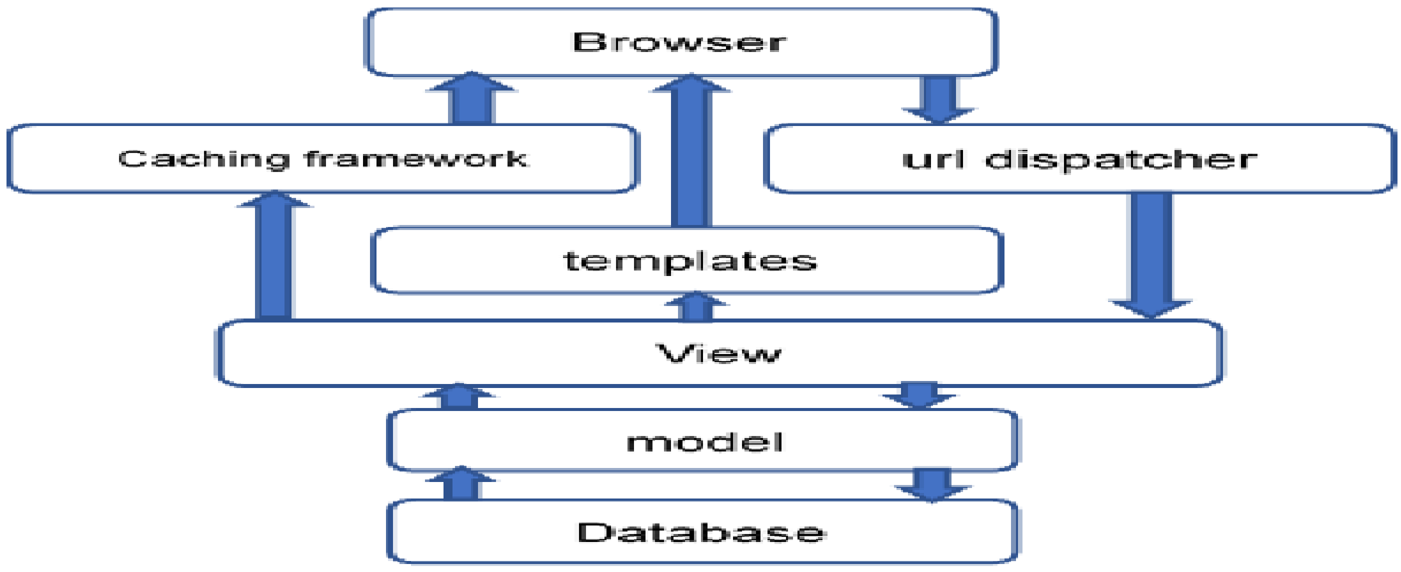


FIG :DJANGO-USER RELATION

5.1.10 CREATE A PROJECT:

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure −

```
myproject/
manage.py
myproject/
_ init _.py
settings.py
urls.py
wsgi.py
```

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

```
$ python manage.py help
```

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

init.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

```
DEBUG = True
```

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.sqlite3',
'NAME': 'database.sql',

'USER': '',5.1.10

'PASSWORD': '',

'HOST': '',

'PORT': '',

}

}
```

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)

PostGreSQL (django.db.backends.postgresql_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/ Quit

the server with CONTROL-C.
```

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

5.1.11 CREATE AN APPLICATION:

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

```
$ python manage.py startapp myapp
```

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

```
myapp/
   _init_____.py
   admin.py
   models.py
   tests.py
   views.py
```

init.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
```

```
from    django    import    forms

class LoginForm(forms.Form):

user = forms.CharField(max_length = 100)

 password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-

 from myapp.forms import LoginForm

 def login(request):

 username = "not logged in"

 if request.method == "POST":

 #Get the posted form

 MyLoginForm = LoginForm(request.POST)

 if MyLoginForm.is_valid():

 username    =    MyLoginForm.cleaned_data['username']

 else:

 MyLoginForm = Loginform()

 return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
 <html>

 <body>

 <form name = "form" action = "{% url "myapp.views.login" %}"

  method = "POST" >{% csrf_token %}

  <div style = "max-width:470px;">
 <center>
  <input type = "text" style = "margin-left:20%;"
   placeholder = "Identifiant" name = "username" />
  </center>
  </div>
  <br>
  <div style = "max-width:470px;">
  <center>
  <input type = "password" style = "margin-left:20%;"
  placeholder = "password" name = "password" />
  </center>
  </div>
  <br>
  <div style = "max-width:470px;">
  <center>
  <button style = "border:0px; background-color:#4285F4; margin-top:8%;
  height:35px; width:80%;margin-left:19%;" type = "submit"
  value = "Login" >
  <strong>Login</strong>
  </button>
  </center>
  </div>
  </form>
  </body>
  </html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>

<body>

You are : <strong>{{username}}</strong>

</body>

</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

 url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),

  url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered −

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −

'django.contrib.sessions.middleware.SessionMiddleware'

And INSTALLED_APPS should have −

'django.contrib.sessions'

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side

```
def login(request):

username = 'not logged in'

if request.method == 'POST':
 MyLoginForm = LoginForm(request.POST)
 if MyLoginForm.is_valid():
username = MyLoginForm.cleaned_data['username']
request.session['username'] = username
else:
MyLoginForm = LoginForm()
return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
if request.session.has_key('username'):
username = request.session['username']
return render(request, 'loggedin.html', {"username" : username})
else:
return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
```

6.SYSTEM DESIGN

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. **GOALS:**

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.

System design interviews are often intimidating. It could be as vague as “designing a well-known product X?”. The questions are ambiguous and seem unreasonably broad. Your weariness is understandable. After all, how could anyone design a popular product in an hour that has taken hundreds if not thousands of engineers to build?

The good news is that no one expects you to. Real-world system design is extremely complicated. Fo example, Google search is deceptively simple; however, the amount of technology that underpins tha simplicity is truly astonishing. If no one expects you to design a real-world system in an hour, what is th benefit of a system design interview?

The system design interview simulates real-life problem solving where two co-workers collaborate on an ambiguous problem and come up with a solution that meets their goals. The problem is open-ended, and there is no perfect answer. The final design is less important compared to the work you put in the design process. This allows you to demonstrate your design skill, defend your design choices, and respond to feedback in a constructive manner.

5.1 SYSTEM ARCHITECTURE

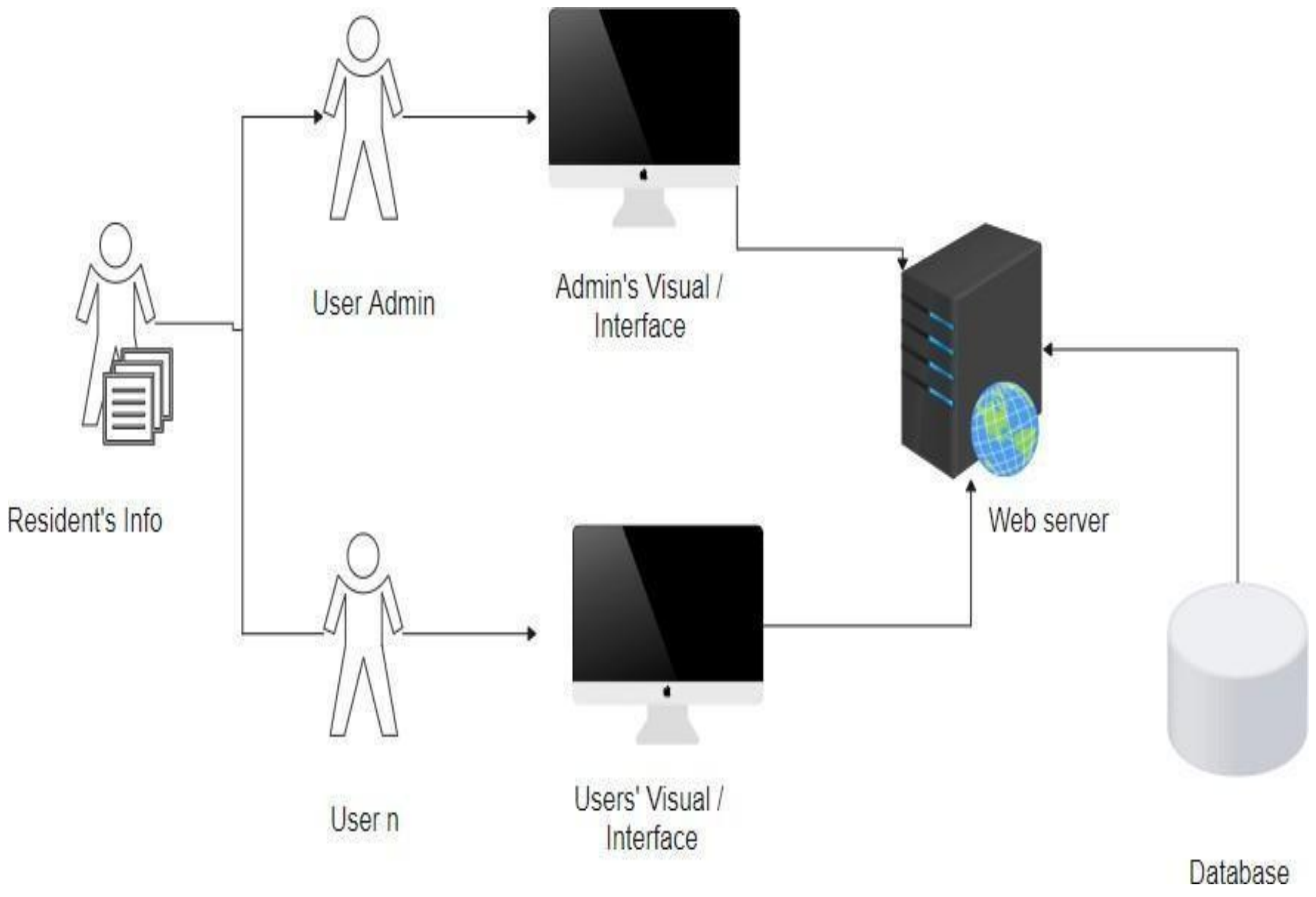


Fig :SYSTEM ARCHITECTURE

System architecture conveys the informational content of the elements consisting of a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that an architecture description may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations

A system architecture primarily concentrates on the internal interfaces among the system's components or subsystem , and on the interface(s) between the system and its external environment, especially the user (In the specific case of computer systems, this latter, special, interface is known as the computer human interface , aka human computer interface, or HCI; formerly called the man- machine interface.)

6.2 DATA FLOW DIAGRAM

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

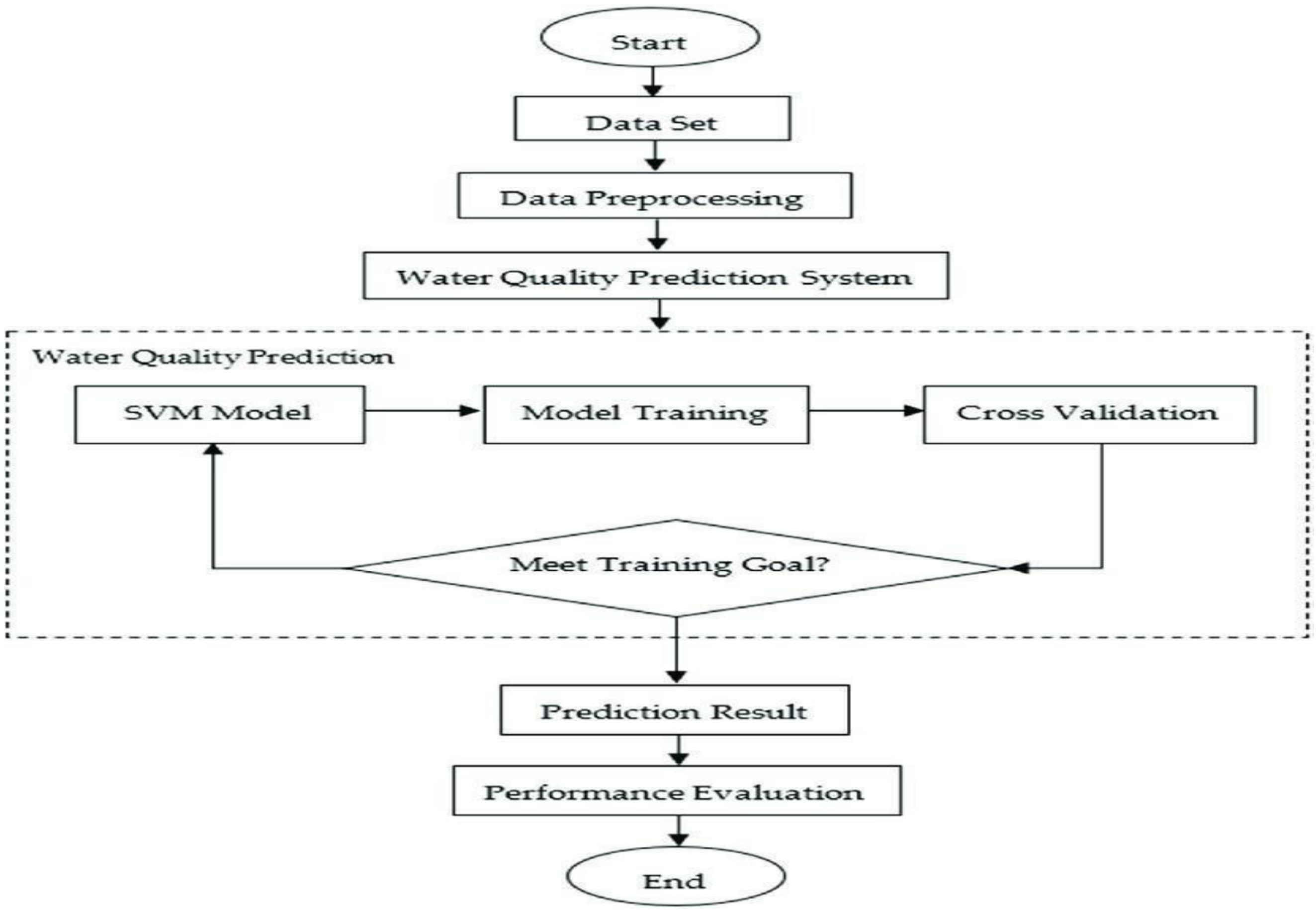


Fig :WORKFLOW OF DATA FLOW

6.3 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

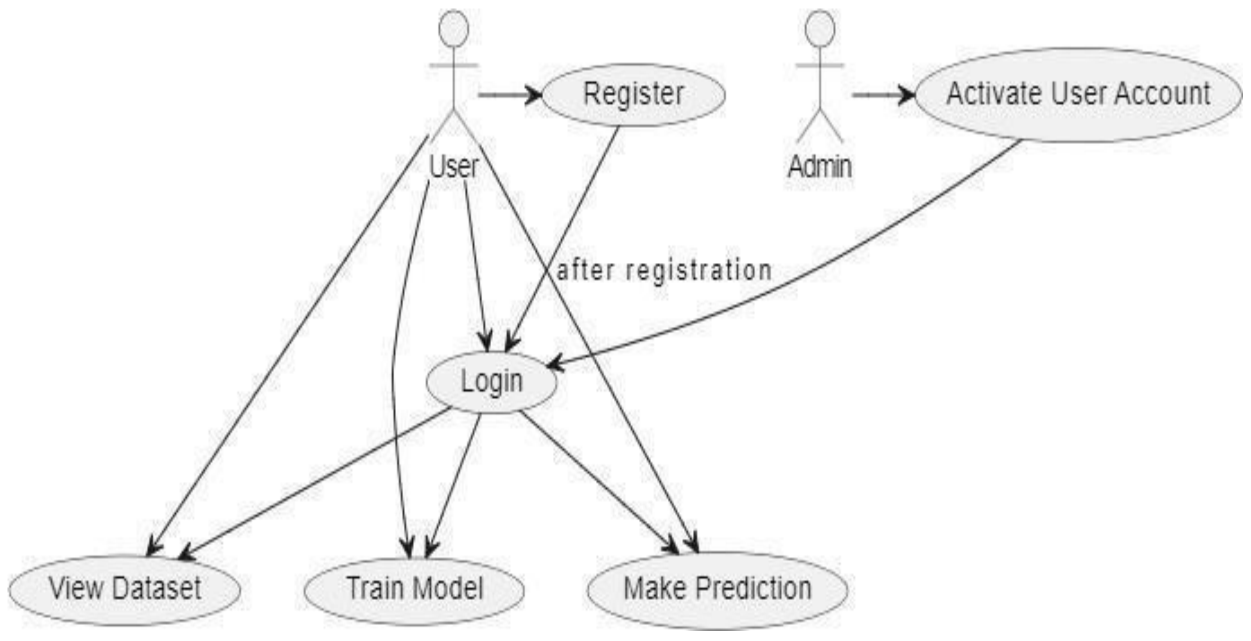


Fig :USE CASE DIAGRAM

Use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A Use Case Diagram Is Usually Simple. It Does Not Show The Detail Of The Use Cases:

- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

As said, a use case diagram should be simple and contains only a few shapes. If yours contain more than 20 use cases, you are probably misusing use case diagram.

Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It iseffective technique for communicating system behavior in the user's terms by specifying all externally

6.4 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

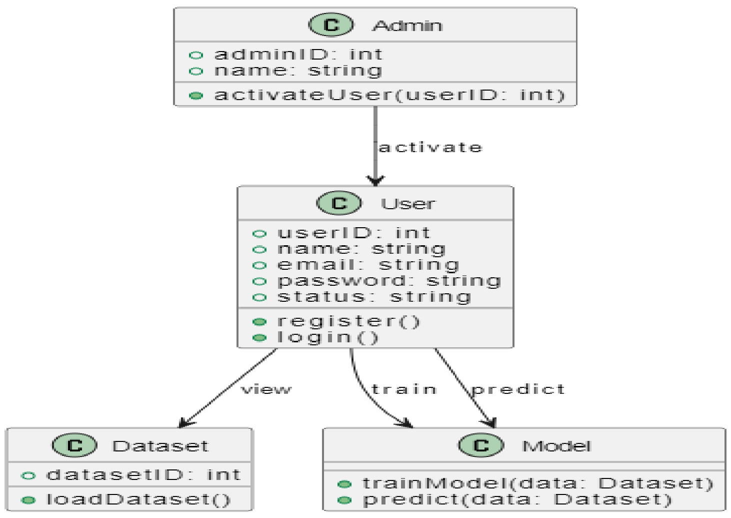


Fig :CLASS DIAGRAM

A description of a group of objects all with similar roles in the system, which consists of:

- Structural features (attributes) define what objects of the class "know"
- Represent the state of an object of the class
- Are descriptions of the structural or static features of a class
- Behavioral features (operations) define what objects of the class "can do"
- Define the way in which objects may interact
- Operations are descriptions of behavioral or dynamic features of a class

Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and relationships between objects. With our UML diagramming software, creating these diagrams is not as overwhelming as it might appear. This guide will show you how to understand, plan, and create your own class diagrams.

6.5 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

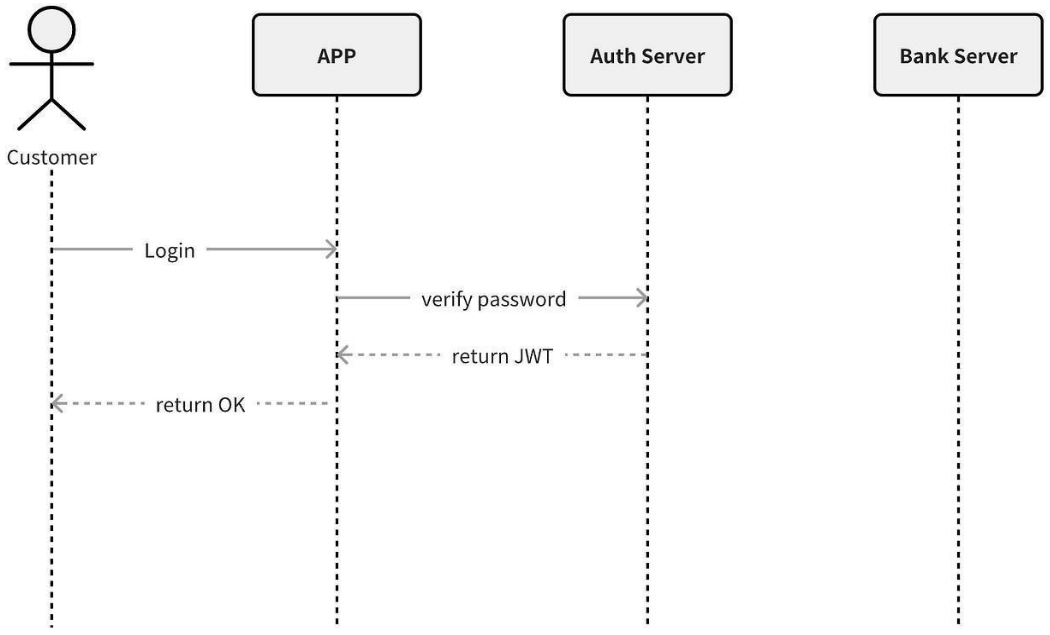


Fig : SEQUENCE DIAGRAM

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagrams

useful to communicate how the business currently works by showing how various business objects interact.Besides documenting an organization's current affairs, a business-level sequence diagram can be used as a requirements document to communicate requirements for a future system implementation. During the requirements phase of a project, analysts can take use cases to the next level by providing a more formal

level of refinement. When that occurs, use cases are often refined into one or more sequence diagrams.An organization's technical staff can find sequence diagrams useful in documenting how a future system should behave. During the design phase, architects and developers can use the diagram to force out the system's object interactions, thus fleshing out overall system design.

6.6 ACTIVITY DIAGRAM:

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

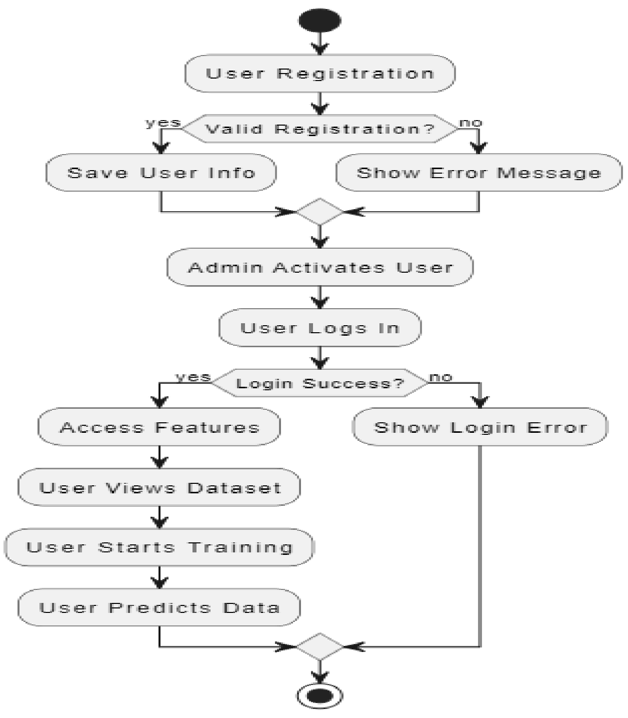


Fig: ACTIVITY DIAGRAM

In UML, an activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process. Activity diagrams are similar to flowcharts because they show the flow between the actions in an activity; however, activity diagrams can also show parallel or concurrent flows and alternate flows.In activity diagrams, you use activity nodes and activity edges to model the flow of control and data between actions.

Activity diagrams are helpful in the following phases of a project:

- Before starting a project, you can create activity diagrams to model the most important workflows.
- During the requirements phase, you can create activity diagrams to illustrate the flow of events that the use cases describe.
- During the analysis and design phases, you can use activity diagrams to help define the behavior of operations.

7. INPUT AND OUTPUT DESIGN

➢ INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

➢ Objectives

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

➢ OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information.
- Create document, report, or other formats that contain information produced by the system.
- The output form of an information system should accomplish one or more of the following objectives.

➢ What is the difference between output and outcome design

The difference between output and outcome in design revolves around the focus on products versus effects.Output in design refers to the immediate tangible and intangible products delivered at the end of the design process. These include prototypes, technical drawings, CAD files, and other deliverables directly produced as a result of design activities. Outputs are the direct results of the inputs and actions taken during the design phase. Outcome, on the other hand, refers to the longer-term effects and impacts of the design outputs. Outcomes focus on how the outputs affect user experience, market performance, and fulfillment of user needs and requirements. They assess the effectiveness of the design in solving a problem or meeting specific goals and can include enhanced user satisfaction, increased sales, or improved operational efficiency.

Design inputs are the king of medical device product development.If a product that is in the market has issues, odds are the issue can be traced back to the design inputs defined during product development.

Design inputs are the foundation of a medical device, and your device is only as effective as the inputs used to create it. Well established design inputs can make the rest of medical device product development easier as a result.Once you've defined design inputs, you are ready to engage in core development. In my opinion, this is one of the most enjoyable aspects of medical device product development.Core development is the stage in which you'll be creating your device prototypes and bench testing them. This process leads to establishing design outputs, which define the medical device components and how it will be assembled. Upon defining design inputs and design outputs, this is the part where your medical device will really start to come to life.

8. MODULES

➢ USER MODULE

The User Module in this project focuses on user management, facilitating tasks like registration, login, and access to core functionalities such as dataset viewing, training, and prediction. A user must first register, providing essential details like their name, email, and password. Upon successful registration, the user account remains inactive until the admin grants activation. Once activated, users can log in to the system using their credentials. After login, users are granted access to various features, such as viewing the dataset and initiating machine learning training or making predictions. Each user's interaction is authenticated, ensuring secure access to data and operations. This module plays a crucial role in enabling users to engage with the machine learning model for training and predictions while maintaining secure access control.

➢ ADMIN MODULE

The Admin Module is responsible for managing user accounts and overseeing system operations. Admins have the exclusive ability to activate user accounts after registration, thereby controlling access to the system's functionalities. Without activation, users cannot log in or utilize features such as training and prediction. The admin can also monitor system activity, manage user permissions, and ensure that only authorized users gain access to critical features. The admin's role in the system is pivotal for maintaining security and ensuring that only vetted users can interact with the machine learning processes, thereby reducing unauthorized access risks.

➢ MACHINE LEARNING MODULE

The Machine Learning Module is the core engine of the project, facilitating the training of predictive models and generating predictions based on user input. Once a user initiates training, the system preprocesses the dataset, cleaning and transforming the data to suit the requirements of machine learning algorithms. In this project, models such as Linear Regression are trained on the provided dataset to predict car prices. Users can trigger this training process from the interface, and the module handles data splitting, model fitting, and evaluation, generating performance metrics such as Mean Squared Error (MSE) and R-squared values. Additionally, once the model is trained, users can input new data to generate predictions. This module ensures that machine learning operations run efficiently, providing accurate results that inform decision-making processes.These module descriptions outline the responsibilities and importance of each component in your system, emphasizing their roles in achieving the overall functionality.

A module is a distinct assembly of components that can be easily added, removed or replaced in a larger system. Generally, a module is not functional on its own. In computer hardware, a module is a component that is designed for easy replacement. In computer software a module is an extension to a main program dedicated to a specific function. In programming, a module is a section of code that is added in as a whole or is designed for easy reusability.

➢ WHAT ARE HARDWARE MODULES

For hardware, a module is an assembly of parts designed to be added and removed from a larger system easily. An example of a hardware module is a stick of RAM Most modules are not functional on their own. They need to be connected to a larger system or be part of a system made up of several modules.The concept of modularity or being made up of modules is common in computer hardware.

➢ A MODULAR SYSTEM HAS MANY ADVANTAGES

Allowing for ease of repair, upgrade and extension of functionality. The various modules can be swapped out as needed.Standardization of hardware as interconnects enables several vendors to produce modules creating better choice for consumers.

➢ WHAT ARE COMPUTER PROGRAMMING MODULES

Modular programming seeks to create separate portions of a program with isolated functionality. These smaller portions can be coded by different teams and then put together into the larger program. These may be called modules, assemblies or packages. In the Java programming language, a module is a distribution format combining the packages and how they are distributed.

These modules are often designed to be reusable. These can be called by the program as needed or used for different areas of the program. A program may import external modules programmed by others. These may also be called a library. Using modules maintained by others is increasingly becoming an attack vector for introducing malware through Supply chain attacks

9. SYSTEM 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. There are various types of test. Each test type addresses a specific testing requirement.

➢ 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.

➢ INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they 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.

➢ FUNCTIONAL TEST

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

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. 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.

➢ 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 configurationoriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **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.

- **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, as most other kinds of 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.

- **UNIT TESTING**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

9.7.1.1 All field entries must work properly.
9.7.1.2 Pages must be activated from the identified link.
9.7.1.3 The entry screen, messages and responses must not be delayed.

Features to be tested

9.7.1.4 Verify that the entries are of the correct format
9.7.1.5 No duplicate entries should be allowed
9.7.1.6 All links should take the user to the correct page.

- **INTEGRATION TESTING**

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.

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.

System testing is a level of software testing that evaluates the complete and integrated software system to verify that it meets specified requirements.

System testing is performed after integration testing and involves testing the entire application as a whole to ensure that all components work together correctly. It focuses on validating the system's compliance with functional and non functional requirement, such as performance, usability, and security. This type of testing simulates real-world scenarios to identify defects and confirm that the software behaves as expected in a production-like environment, ultimately ensuring the quality and reliability of the system before it is deployed.

You must verify the software requirements, architecture, design, codes, etc., and meet the business requirements and standards. Also, you should verify these software artifacts are error-free and built according to the specifications.

- System testing covers the end-to-end functions of a system, and thus it provides reliability to the system.
- It helps to solve bugs after post-production.
- It tests the entire system architecture as per the business requirement.
- This testing keeps the new and previous functionalities in a single system to help the user understand the benefits of the newly added features.

SCOPE OF SYSTEM TESTING

The scope of system testing includes the following key areas:

- **Functional testing** : This ensures that the system meets its requirements and performs the intended functions as specified.

- **Non functional testing** : This aspect verifies the system’s non-functional requirements, including performance, reliability, usability, and security.

- **Interface Testing**: This checks that the system properly interfaces with other systems and components, ensuring seamless interaction.

- **Stress Testing**: This assesses the system’s ability to handle high traffic or load, identifying how it performs under pressure.

- **Recovery Testing**: This evaluates the system’s capability to recover from failures or errors, ensuring stability and reliability.

While the specific tests conducted during system testing may vary depending on the system, the primary goal is to confirm that the system is ready for end users.

System Testing ProcessSystem testing is a critical phase in the software development lifecycle that ensures the complete application functions as intended before release. It involves validating both functional and non-functional requirements through a structured process that includes planning, design, execution, and closure.

- **Test planning**

This phase involves defining the scope, objectives, resources, schedule, and testing approach for the system testing process. It sets the groundwork for the entire testing effort.
Example: A team outlines a test plan for an e-commerce application, specifying the types of tests to be performed, such as functional, performance, and security testing, along with timelines and team responsibilities

- **Test Design**:

During this phase, testers create detailed test cases based on the requirements and specifications. Test scenarios are designed to cover all aspects of the system, including edge cases.
Example: For a user login feature, test cases might include scenarios for valid and invalid credentials,

- **Teat Environment Set Up**

In this step, the testing environment is prepared to mirror the production environment as closely as possible. This includes configuring hardware, software, and network settings.

Example: Setting up a staging server that replicates the production database and application configuration to test the new features before going live.

- **Test Execution**

Testers execute the designed test cases in the prepared environment, recording the results and any defects discovered during the process. This phase may involve manual testing or automated scripts.

Example: A tester logs into the e-commerce application, attempts various login scenarios, and documents any errors encountered, such as incorrect error messages or system crashes.

- **Test Closure**

After testing is complete, the team evaluates the results, reports defects, and assesses whether the system meets the acceptance criteria. A final report is created to summarize the testing efforts and outcomes.

Example: The team compiles a report indicating the number of test cases executed, the number passed/failed, and any critical defects that need resolution before deployment, providing a clear overview of the testing process.

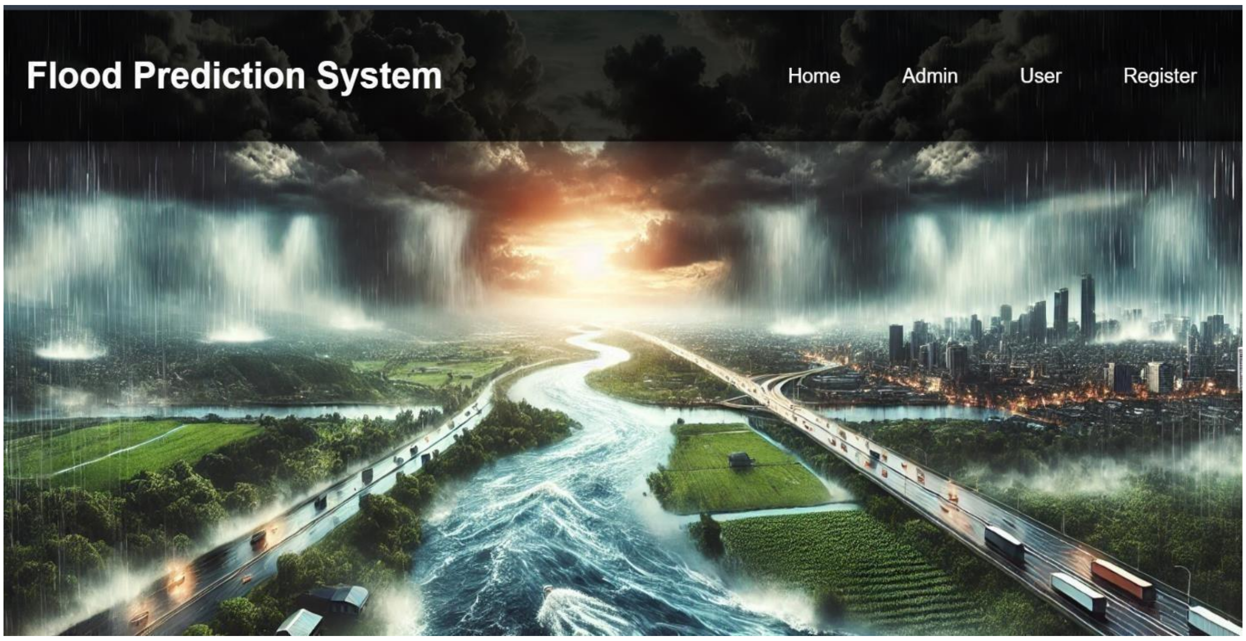
10. SAMPLE TEST CASES

Test Case Description	Test Steps	Expected Result	Result	Remarks (If Fail)
1.User Registratin	1. Navigate to registration page. 2. Enter valid user details. 3. Submit the registration form.	User is successfully registered and a confirmation message is shown.	Pass/Fail	If fail, check if fields are validated properly or if server errors.
2.Admin User Activation	1. Admin logs in. 2. Navigate to user activation section. 3. Select a pending user. 4. Click on "Activate".	User is successfully activated and their status changes to active.	Pass/Fail	If fail, verify admin rights or database update issues.
3.User Login	1. Navigate to login page. 2. Enter valid login credentials. 3. Click on "Login".	User is logged in successfully and redirected to the homepage.	Pass/Fail	If fail, check if login credentials are correct or user is activated.
4. Accessing Dataset View (Sample Data)	1. Login as an activated user. 2. Navigate to the "Dataset View" section. 3. Click on "Dataset View".	Sample dataset is displayed.	Pass/Fail	If fail, check if dataset is available or correct URL is used.
5. Accessing Preprocesse Data	1. Login as an activated user. 2. Navigate to the "Preprocessed Data" section. 3. Click on "Preprocessed Data".	Preprocessed data is displayed successfully.	Pass/Fail	If fail, check if preprocessing function is working and data is available.

TABLE:SAMPLE TEST CASE

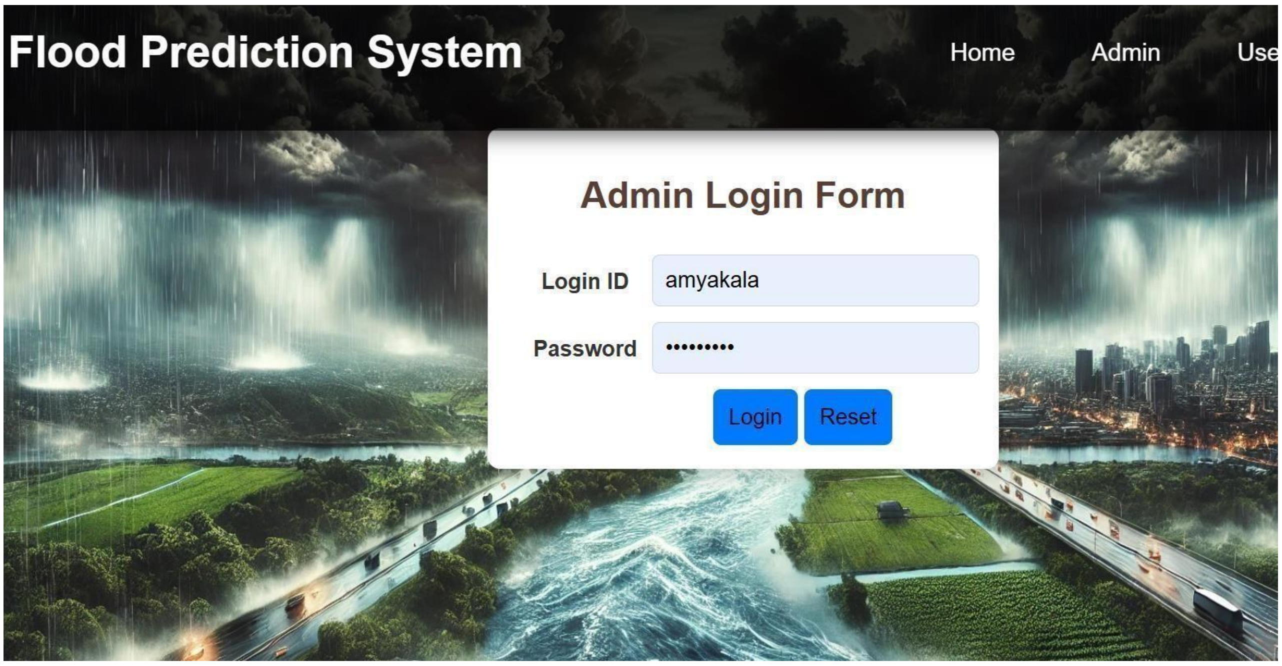
11. SCREENSHOTS

Home page:



SCREEN SHOT 12.1
Home page

Admin login:

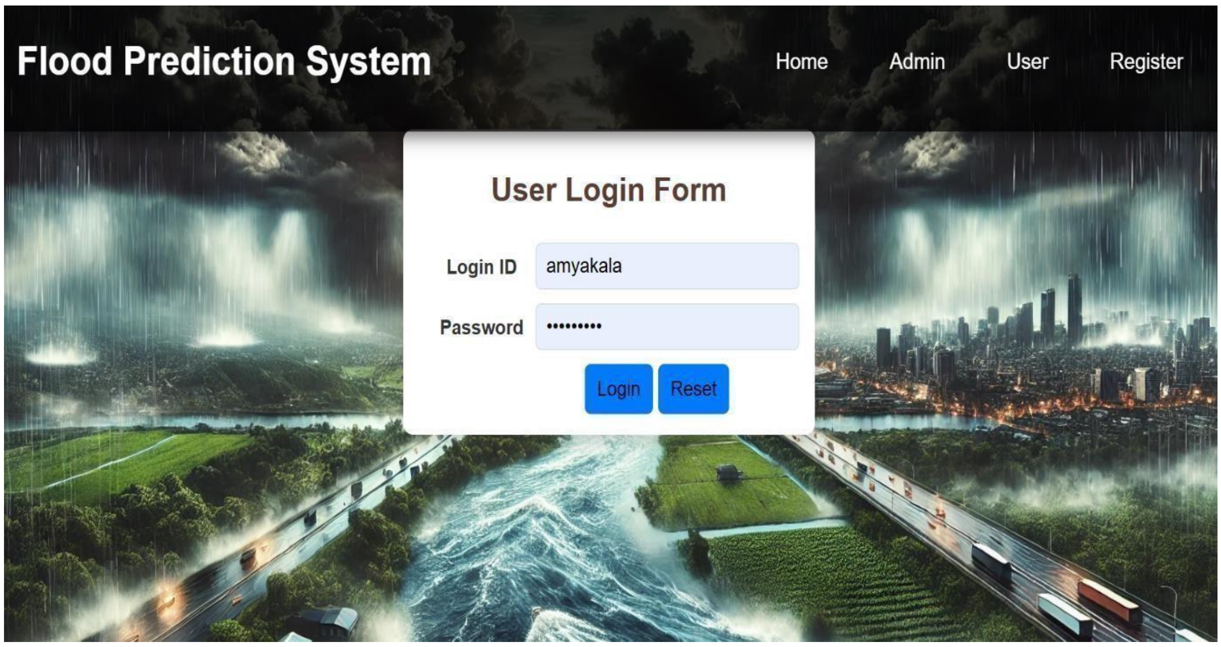


Login ID amyakala

Password •••••••••

SCREEN SHOT 12.2
Admin page

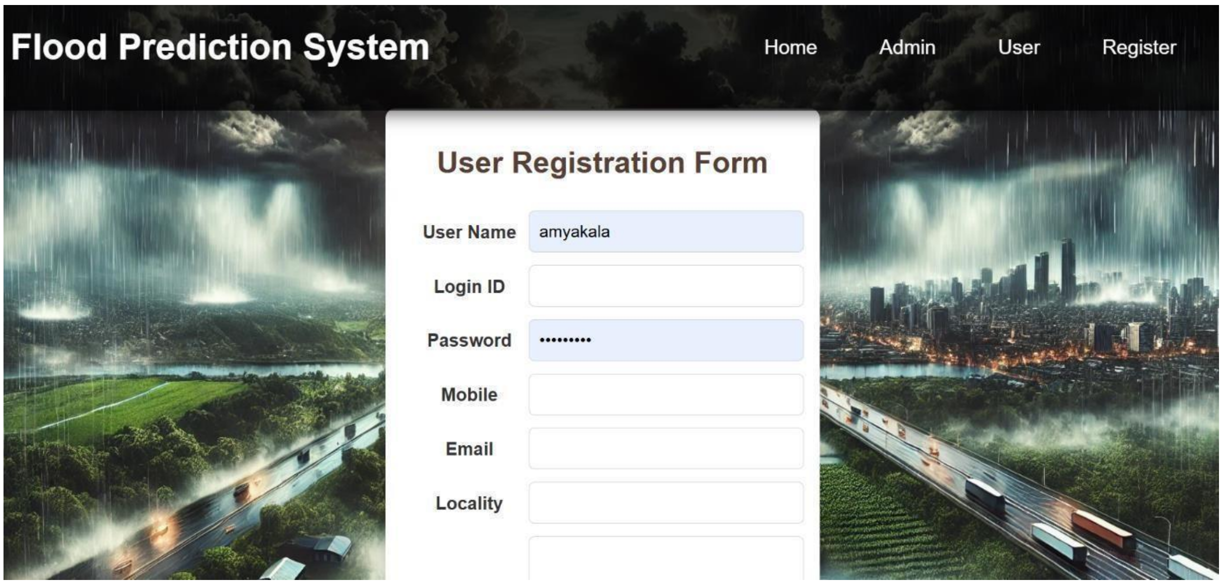
User login:



SCREEN SHOT 12.3

User Login

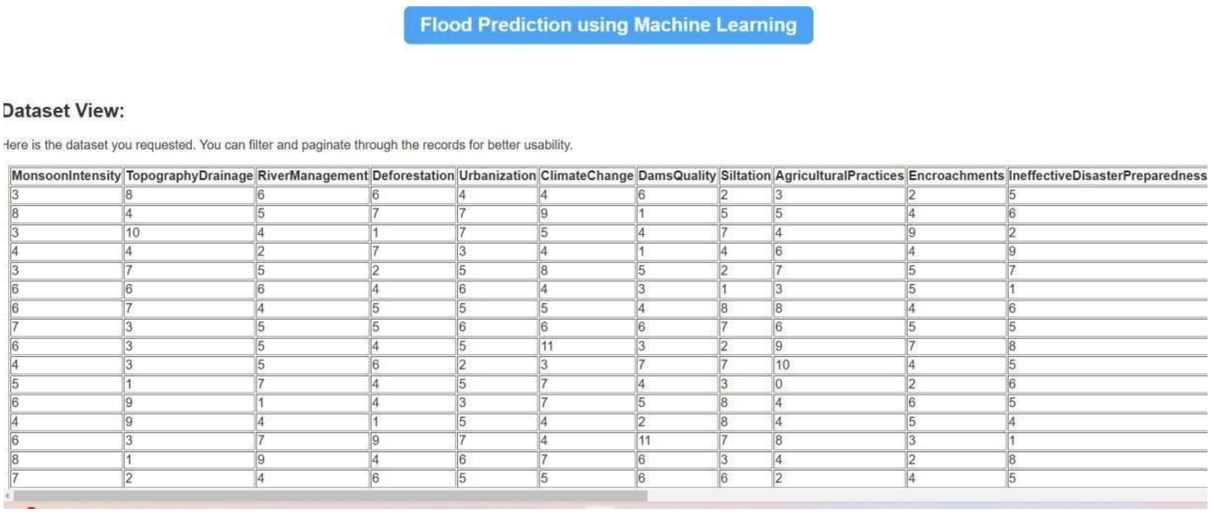
User registration:



SCREEN SHOT 12.4

User Registration

Data Set:



Flood Prediction using Machine Learning

Dataset View:

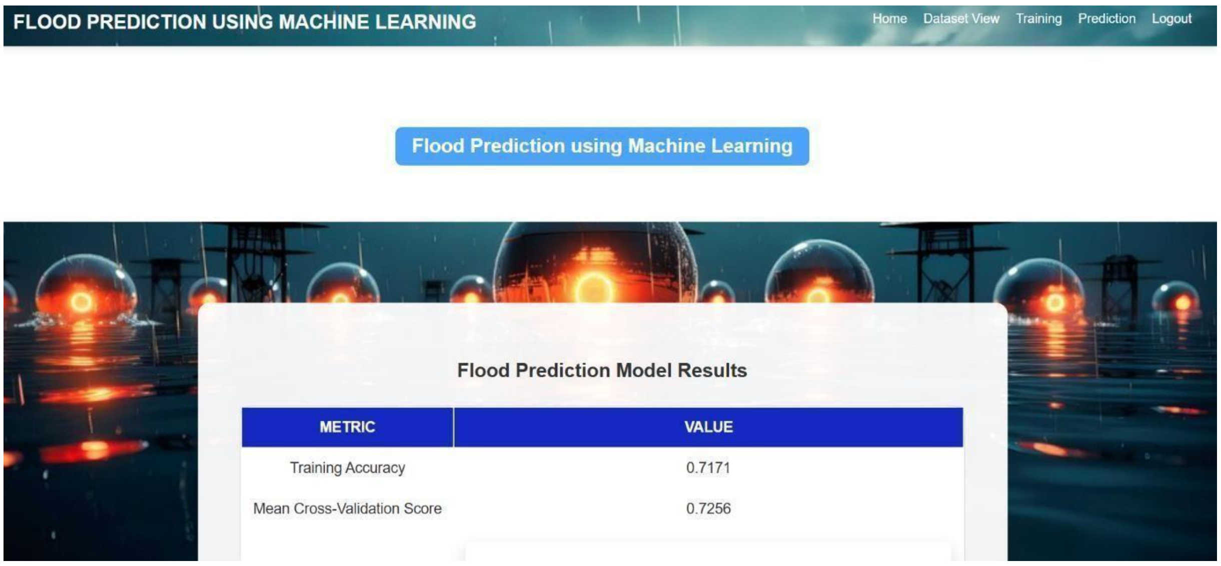
Here is the dataset you requested. You can filter and paginate through the records for better usability.

MonsoonIntensity	TopographyDrainage	RiverManagement	Deforestation	Urbanization	ClimateChange	DamsQuality	Siltation	AgriculturalPractices	Encroachments	IneffectiveDisasterPreparedness
3	8	6	6	4	4	6	2	3	2	5
8	4	5	7	7	9	1	5	5	4	6
3	10	4	1	7	5	4	7	4	9	2
4	4	2	7	3	4	1	4	6	4	9
3	7	5	2	5	8	5	2	7	5	7
6	6	6	4	6	4	3	1	3	5	1
6	7	4	5	5	5	4	8	8	4	6
7	3	5	5	6	6	6	7	6	5	5
6	3	5	4	5	11	3	2	9	7	8
4	3	5	6	2	3	7	7	10	4	5
5	1	7	4	5	7	4	3	0	2	6
6	9	1	4	3	7	5	8	4	6	5
4	9	4	1	5	4	2	8	4	5	4
6	3	7	9	7	4	11	7	8	3	1
8	1	9	4	6	7	6	3	4	2	8
7	2	4	6	5	5	6	6	2	4	5

SCREEN SHOT 12.5

Data Set

Training:


FLOOD PREDICTION USING MACHINE LEARNING
Home
Dataset View
Training
Prediction
Logout
Flood Prediction using Machine Learning

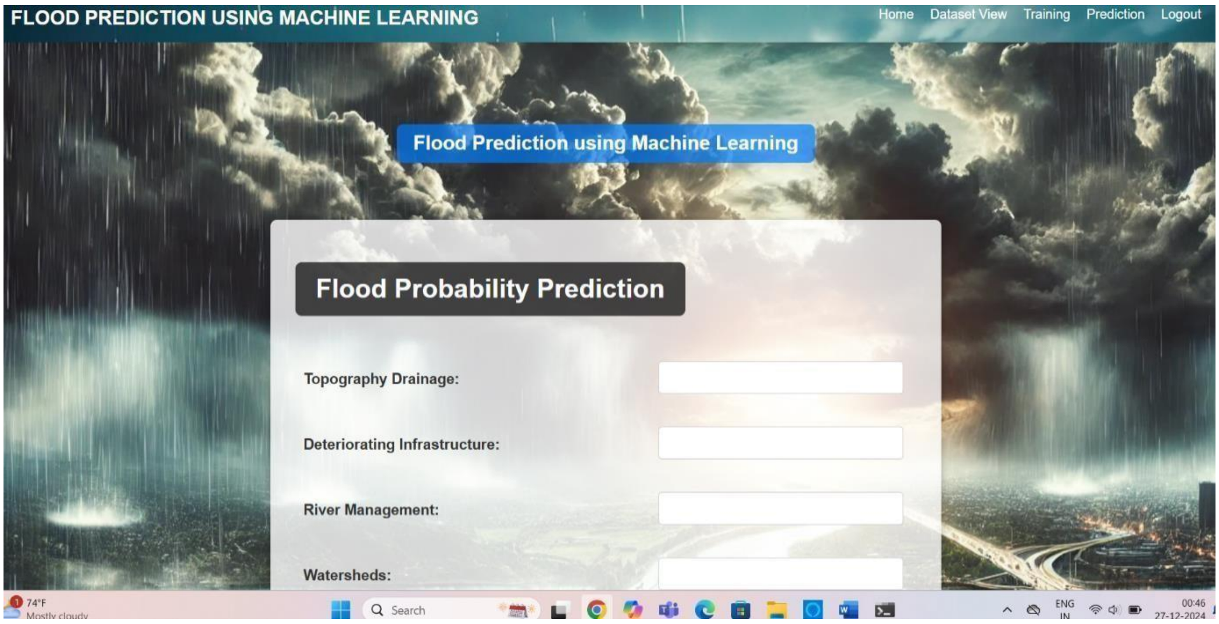
Flood Prediction Model Results

METRIC	VALUE
Training Accuracy	0.7171
Mean Cross-Validation Score	0.7256

SCREEN SHOT 12.6

Training

Prediction:


FLOOD PREDICTION USING MACHINE LEARNING
Home
Dataset View
Training
Prediction
Logout
Flood Prediction using Machine Learning

Flood Probability Prediction

Topography Drainage:

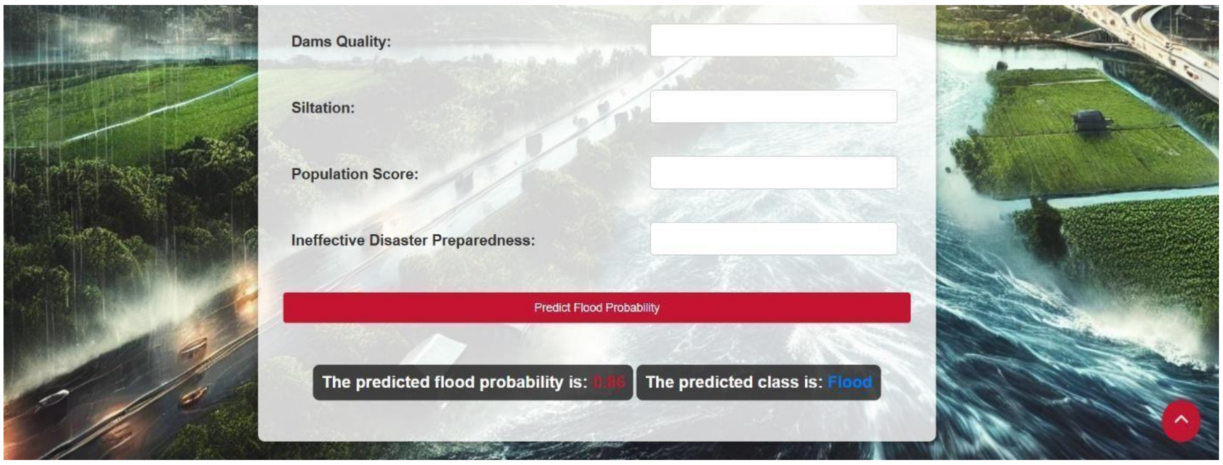
Deteriorating Infrastructure:

River Management:

Watersheds:

SCREEN SHOT 12.7

Prediction



Dams Quality:

Siltation:

Population Score:

Ineffective Disaster Preparedness:

The predicted flood probability is: 0.86 The predicted class is: Flood

SCREEN SHOT 12.7
Prediction

12. FUTURE ENCHACEMENT

Incorporation of Additional Features:

Future development can integrate more variables such as satellite data, real-time weather forecasts, soil moisture levels, and historical flood events. These additional features will improve prediction accuracy by accounting for a wider range of factors influencing flood occurrences.

Use of Advanced Machine Learning Algorithms:

Implementing more advanced models such as Random Forest, Gradient Boosting, XGBoost, and deep learning algorithms (e.g., Convolutional Neural Networks) will help capture more complex patterns in flood data, leading to more accurate and reliable predictions.

Mobile Application Development:

Developing a mobile application for the flood prediction system will make it more accessible to local authorities, government agencies, and the general public. This will enable users to receive real-time flood alerts and predictions on their smartphones, improving flood preparedness and response times.

13. CONCLUSION

This research has successfully developed a robust flood prediction model using various machine learning techniques, including Logistic Regression, Random Forest, ExtraTree, XGBoost, LGBM, and CatBoost classifiers. By utilizing historical data and key flood-related attributes such as rainfall, water levels, and other environmental factors, the model provides more accurate and data-driven flood predictions.

The study highlights the limitations of traditional flood prediction methods, which often rely on predefined thresholds and may lack the adaptability and accuracy needed for real-time predictions. The proposed machine learning approach significantly enhances the prediction process, allowing for better flood forecasting and more reliable disaster management.

The results demonstrate that Logistic Regression outperforms other algorithms in terms of accuracy and performance for this particular dataset, making it the most suitable model for flood prediction. This approach allows for more efficient and scalable flood forecasting, which can be beneficial for both government agencies and the general public in flood risk management.

In conclusion, this system addresses the complexity of flood prediction by improving accuracy and providing timely forecasts. Future work could focus on incorporating additional variables, integrating real-time data, and exploring more advanced machine learning techniques to further enhance the robustness of the model. With continuous development, the system will remain adaptable to evolving flood patterns and help in minimizing the risks associated with floods.

14. SOURCE CODE

```
from   pyexpat.errors   import   messages

from   django.shortcuts   import   render

from   django.contrib   import   messages

from Flood_Prediction import settings

from   users.forms   import   UserRegistrationForm

from users.models import UserRegistrationModel

def UserRegisterActions(request):

if request.method == 'POST':

form = UserRegistrationForm(request.POST)

if form.is_valid():

print('Data  is  Valid')

form.save()

messages.success(request,  'You  have  been  successfully  registered')

form = UserRegistrationForm()

return  render(request,  'UserRegistrations.html',  {'form':  form})

else:

 messages.success(request,     'Email     or     Mobile     Already     Existed')

 print("Invalid form")

  else:

 form = UserRegistrationForm()

 return  render(request,  'UserRegistrations.html',  {'form':  form})

 def UserLoginCheck(request):

 if request.method == "POST":

 loginid = request.POST.get('loginid')

 pswd = request.POST.get('pswd')

 print("Login ID = ", loginid, ' Password = ', pswd)

        try:
```

```
check   =   UserRegistrationModel.objects.get(
loginid=loginid, password=pswd)
status      =      check.status
print('Status is = ', status)
if status == "activated":
request.session['id']              =              check.id
request.session['loggeduser']   =   check.name
request.session['loginid']           =           loginid
request.session['email']       =       check.email
print("User id At", check.id, status)
return render(request, 'users/UserHomePage.html', {})
else:
messages.success(request,  'Your  Account  Not  at  activated')
return render(request, 'UserLogin.html')
except Exception as e:
print('Exception  is  ',  str(e))
pass
messages.success(request,  'Invalid  Login  id  and  password')
return render(request, 'UserLogin.html', {})
def UserHome(request):
return render(request, 'users/UserHomePage.html', {})


from    django.conf    import    settings
import pandas as pd
from pathlib import Path
from django.shortcuts import render
 def DatasetView(request):
path = Path(settings.MEDIA_ROOT) / 'flood (2).csv'
```

```
      try:

      df     =    pd.read_csv(path)

      except FileNotFoundError:

      return render(request, 'users/viewdataset.html', {'data': '<p style="color: red;">CSV file not found.
Please upload the file to the media folder.</p>'})

     except Exception as e:

     return render(request, 'users/viewdataset.html', {'data': f'<p style="color: red;">Error reading CSV
file: {e}</p>'})

         df_html = df.to_html(classes='table table-striped table-bordered', index=False)  # Added classes
for styling

    return              render(request,              'users/viewdataset.html',              {'data':              df_html})

######################################################################

import pandas as pd

import numpy as np

from django.shortcuts import render

from  sklearn.model_selection  import  train_test_split,  GridSearchCV,  cross_val_score

from sklearn.linear_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from  sklearn.metrics  import  accuracy_score,  confusion_matrix

import seaborn as sns

import    matplotlib.pyplot    as    plt

import joblib

import os

# Training view for Flood Probability Prediction

def training(request):

# Load the dataset

data_path = os.path.join(settings.MEDIA_ROOT, 'flood (2).csv')  # Assuming flood.csv is uploaded

a1 = pd.read_csv(data_path)

   print(a1)  # Print the dataset for debugging purposes (can be removed in production)
```

```
# Step 1: Compute and display the correlation matrix
correlation_matrix = a1.corr()
print(correlation_matrix)  # Print the correlation matrix for debugging purposes (optional)
# Step 2: Prepare the data
X = a1[['TopographyDrainage', 'DeterioratingInfrastructure', 'RiverManagement', 'Watersheds',
'DamsQuality', 'Siltation', 'PopulationScore', 'IneffectiveDisasterPreparedness']]
y = (a1['FloodProbability'] > 0.5).astype(int)  # Binarize target variable
# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
# Step 3: Define parameter grid for hyperparameter tuning
param_grid = {
'C': [0.1, 1, 10, 100, 1000],
'solver': ['liblinear', 'saga'],
'penalty': ['l2']
}
# Step 4: GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(LogisticRegression(max_iter=1000), param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
# Step 5: Use the best model found by GridSearchCV
best_model = grid_search.best_estimator_
# Step 6: Evaluate model performance
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
cv_scores = cross_val_score(best_model, X_scaled, y, cv=5, scoring='accuracy')
# Confusion matrix
```

```
 conf_matrix = confusion_matrix(y_test, y_pred)

 plt.figure(figsize=(6, 5))

 sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=np.unique(y),
yticklabels=np.unique(y))

plt.title("Confusion            Matrix")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.savefig(os.path.join(settings.MEDIA_ROOT,    'confusion_matrix.png'))

plt.close()

 # Correlation matrix visualization

plt.figure(figsize=(8, 6))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title("Correlation Matrix")

 plt.savefig(os.path.join(settings.MEDIA_ROOT, 'correlation_matrix.png'))

plt.close()

#      Save      the      model      and      scaler

joblib.dump(best_model,         'best_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

context        =        {

'accuracy': accuracy,

'cv_scores':                         cv_scores,

'mean_cv_score': cv_scores.mean(),

'confusion_matrix_plot':           'confusion_matrix.png',

'correlation_matrix_plot':        'correlation_matrix.png',

'MEDIA_URL': settings.MEDIA_URL,

}

return render(request, 'users/training.html', context)

# Prediction view for Flood Probability Prediction

def prediction(request):
```

```
# Load the model and scaler
if not os.path.exists('best_model.pkl') or not os.path.exists('scaler.pkl'):
return render(request, 'users/predictForm.html', {'error': 'Model has not been trained yet.'})
model = joblib.load('best_model.pkl')
scaler = joblib.load('scaler.pkl')
if request.method == 'POST':
# Collect user input
user_input = {
'TopographyDrainage': float(request.POST.get('TopographyDrainage', 0)),
'DeterioratingInfrastructure':  float(request.POST.get('DeterioratingInfrastructure',  0)),
'RiverManagement': float(request.POST.get('RiverManagement', 0)),
'Watersheds': float(request.POST.get('Watersheds', 0)),
'DamsQuality': float(request.POST.get('DamsQuality', 0)),
'Siltation': float(request.POST.get('Siltation', 0)),
'PopulationScore': float(request.POST.get('PopulationScore', 0)),
'IneffectiveDisasterPreparedness': float(request.POST.get('IneffectiveDisasterPreparedness', 0)),
}
# Convert input to a DataFrame and scale features
input_df          =          pd.DataFrame([user_input])
input_scaled = scaler.transform(input_df)
 # Predict using the model
 predicted_probability = model.predict_proba(input_scaled)[0, 1]
 predicted_class = "Flood" if (predicted_probability > 0.5).astype(int) == 1 else "No Flood"
 return render(request, 'users/predictForm.html', {
 'user_input':                              user_input,
 'predicted_probability': predicted_probability,
 'predicted_class': predicted_class, # Updated to return "Flood" or "No Flood"
   })
 return     render(request,     'users/predictForm.html')
 from sklearn.model_selection import train_test_split
 from  sklearn.preprocessing  import  OneHotEncoder
 from  sklearn.linear_model  import  LinearRegression
 from sklearn import metrics
 from sklearn.metrics import classification_report
# Create your views here.
def UserRegisterActions(request):
 if request.method == 'POST':
 form = UserRegistrationForm(request.POST)
  if form.is_valid():
  print('Data is Valid')
    form.save()
   messages.success(request,  'You have been successfully registered')
    form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})
    else:
   messages.success(request,     'Email     or     Mobile     Already     Existed')
    print("Invalid form")
    else:
    form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})
    def UserLoginCheck(request):
    if request.method == "POST":
    loginid     =     request.POST.get('loginid')
    pswd = request.POST.get('pswd')
    print("Login ID = ", loginid, ' Password = ', pswd)
        try:
```

```
check   =   UserRegistrationModel.objects.get(
loginid=loginid, password=pswd)
status     =     check.status
print('Status is = ', status)
if status == "activated":
request.session['id']             =             check.id
request.session['loggeduser']   =   check.name
request.session['loginid']          =          loginid
request.session['email']       =       check.email
print("User id At", check.id, status)
return render(request, 'users/UserHomePage.html', {})
else:
messages.success(request,  'Your  Account  Not  at  activated')
return render(request, 'UserLogin.html')
except Exception as e:
print('Exception  is  ',  str(e))
pass
messages.success(request,  'Invalid  Login  id  and  password')
return render(request, 'UserLogin.html', {})
def UserHome(request):
 return render(request, 'users/UserHomePage.html', {})
def DatasetView(request):
 path = settings.MEDIA_ROOT + "//" + 'used_cars.csv'
 df = pd.read_csv(path)
 df = df.to_html
 return render(request, 'users/viewdataset.html', {'data': df})
```

```
check   =   UserRegistrationModel.objects.get(
loginid=loginid, password=pswd)
status     =     check.status
print('Status is = ', status)
if status == "activated":
request.session['id']             =             check.id
request.session['loggeduser']   =   check.name
request.session['loginid']           =           loginid
request.session['email']       =       check.email
print("User id At", check.id, status)
return render(request, 'users/UserHomePage.html', {})
else:
messages.success(request,  'Your  Account  Not  at  activated')
return render(request, 'UserLogin.html')
except Exception as e:
print('Exception  is  ',  str(e))
pass
messages.success(request,  'Invalid  Login  id  and  password')
return render(request, 'UserLogin.html', {})
def UserHome(request):
 return render(request, 'users/UserHomePage.html', {})
def DatasetView(request):
 path = settings.MEDIA_ROOT + "//" + 'used_cars.csv'
 df = pd.read_csv(path)
 df = df.to_html
 return render(request, 'users/viewdataset.html', {'data': df})
```

15.REFERENCES

[1] [Sarker I H. Furhad M. H and Nowrozy R, “AI-Driven cybersecurity: an overview, security intelligence modeling and research directions” SN COMPUT. SCI. Vol.2, no. 173, 2021. https://doi.org/10.1007/s42979-021-00557-0

[2] Mohssen Mohammed, Muhammad Badruddin Khan, Eihab Bashier and Mohammed Bashier, “Machine learning: algorithms and applications”, CRC Press, 2016. https://doi.org/10.1201/9781315371658

[3] Botvinick M, Ritter S, Wang J.X. Kurth-Nelson Z. Blundell C. and Hassabis D, “ Reinforcement Learning, Fast and Slow’, Trends Cogn. Sci. , vol. 23, no. 5, pp 408–422, 2019.

[4] Taiwo, O. A, “Types of Machine Learning Algorithms, New Advances in Machine Learning” Yagang Zhang (Ed.), ISBN: 978-953-307-034- 6, InTech, University of Portsmouth United Kingdom. Pp 3 – 31, 2010.

[5] K. Vamshi, S. K. S, B. R. Muralidhar, N. Manjunath, and P. Savitha, “A Review on Rainfall Prediction using Machine Learning and Neural Network,” pp. 2763–2769, 2021.

[6] A. O. Hashi, A. A. Abdirahman, M. A. Elmi, and S. Z. Mohd, “A Real- Time Flood Detection System Based on Machine Learning Algorithms with Emphasis on Deep Learning,” vol. 69, no. 5, pp. 249–256, 2021, doi: 10.14445/22315381/IJETT-V69I5P232

[7] Chang, K. Hsu, and L. Chang, “Flood Forecasting Using Machine Learning Methods”, February 2019 , ISBN 978-3-03897-548-9 .

[8] Mosavi, P. Ozturk, and K. W. Chau, “Flood prediction using machine learning models: Literature review,” Water (Switzerland), vol. 10, no. 11, pp. 1–40, 2018, doi: 10.3390/w10111536.

[9] Hosseiny, F. Nazari, V. Smith, and C. Nataraj, “OPEN A Framework for Modeling Flood Depth Using a Hybrid of Hydraulics and Machine Learning,” Sci. Rep., pp. 1–14, 2020, doi: 10.1038/s41598-020-65232- 5

[10] S. Tehrany, B. Pradhan, and M. N. Jebur, “Flood susceptibility mapping using a novel ensemble weights-of-evidence and support vector machine models in GIS,” J. Hydrol., vol. 512, pp. 332–343, May 2014, doi: 10.1016/j.jhydrol.2014.03.008

[11] M. Yaseen and I. Ebtehaj, “Hybrid Data Intelligent Models and Applications for Water Level Prediction,” no. 1, pp. 121–139, doi: 10.4018/978-1- 5225-4766-2.ch006.

[12] T. Bui, T. A. Tuan, H. Klempe, B. Pradhan, and I. Revhaug, “Spatial prediction models for shallow landslide hazards: a comparative assessment of the efficacy of support vector machines, artificial neural networks, kernel logistic regression, and logistic model tree,” Landslides, vol. 13, pp. 361–378, 2015

[13] Han, L. Chan, and N. Zhu, “Flood forecasting using support vector machines,” J.Hydroinformatics, vol. 9, no. 4, pp. 267–276, Oct. 2007, doi: 10.2166/hydro.2007.027

[14] A. Vinothini, L. Kruthiga, and U. Monisha, “Prediction of Flash Flood using Rainfall by MLP Classifier,” no. 1, pp. 425–429, 2020, doi: 10.35940/ijrte.F9880.059120.

[15] Dazzi, R. Vacondio, and P. Mignosa, “ FloodStage Forecasting Using Machine-Learning Methods : A Case Study on the Parma River (Italy),” 2021.

[16] Demaris A, “A Tutorial in Logistic Regression Published by: National Council on Family Relations”, J. Marriage Fam. 1995, 57, 956–968.