AIR AWARE- SMART AIR QUALITY PREDICTION SYSTEM

Presented by : Davire Divija Nandana.

Mentor : Shakthi Gopalakrishnan

Batch Number : 06

Team Number : 01

Team Members: DivijaNandana

- Lokesh
- Rajalakshmi
- Rahul
- Sreya

Day 1

Understanding the Air Aware Project

Air Aware is a smart system that helps us understand how clean or polluted the air around us is. It collects pollution data, studies patterns, and predicts future air quality using machine learning models. The main aim is to create awareness and alert users when the environment becomes unsafe.

Real-Time Air Pollution Data

Real-time data means the system continuously receives updated pollution readings. Some common pollutant indicators are:

- **PM2.5:** Very tiny particles harmful to lungs
- **CO_2:** Carbon dioxide
- **NO_2:** Nitrogen dioxide
- **O_3, SO_2:** Other harmful pollutants

Basic Components of the System

- **Frontend:** The user interface where charts, graphs, and predictions are displayed
- **Backend:** Python logic that processes input and generates predictions
- **Database:** A place where datasets and sensor/dummy data are stored
- **Dataset:** Air quality information from websites like Kaggle or HuggingFace

Project Overview

- Aim is to study pollution in a particular region
- Analyze past data, show present trends, and predict future AQI
- Use models like Linear Regression, Random Forest, and SVM

Objectives for the Project

- Collect datasets containing PM2.5, NO_2, CO_2, and other pollutant values
- Clean and prepare the data before training the model
- Compare predicted values with actual readings

Day 2

Team Formation & UI Planning

Dataset Selection

Our team chose the **Delhi Air Quality Dataset** from Kaggle, which contains pollutant readings for different days and months.

Understanding the Goal

We need to build a system that:

- Predicts pollution levels
- Shows historic AQI trends
- Displays current air quality
- Shows whether pollution will worsen in the coming years
- Helps users understand where the air stands today and where it's heading

Frontend Planning

- React is suggested because it helps create modern, clean, and attractive UI components
- Every team must create their own style of dashboard
- UI should display:
    - Last year's air quality
    - Prediction for coming days
    - Heatmap of model accuracy
    - Graphs of past months and years

Backend Interaction

- Frontend sends **payload** (input) to backend
- Backend processes the input using ML models
- Results are sent back to frontend as **response**
- Data flow should be smooth and well-structured

Future Possibility

If the system is built quickly and accurately, it can later be connected to real sensors to read live pollution levels.

Completion Target

- First 25% of project must be ready by **23rd November**
- Model + dashboard must show at least 3 months of data

Day 3

Libraries, APIs & Backend Concepts

Python Libraries Used

- **Pandas:** For cleaning and analyzing data
- **NumPy:** For numerical operations
- **Matplotlib:** For creating graphs
- **Heatmaps:** Used to show model accuracy visually

Understanding APIs

An API is a bridge that helps the **frontend** and **backend** talk to each other.
Whatever the user enters in the UI is sent to the backend through API, and the backend returns the result.

Fast API

- A modern, fast Python framework
- Supports **async** programming (multiple tasks at once)
- Good for large-scale applications

Flask

- Simple and beginner-friendly
- Works in **sync** mode (one task at a time)
- Easy to understand and use for small to medium projects

GET vs POST

- **GET:** Used to request data
- **POST:** Used to send data to backend (payload in body)

Postman

Postman is a tool that allows us to test APIs even before the frontend is ready.

Important Note

Before building the API, we must know how the frontend will send data.
If the payload format doesn't match, the API will fail.

WebSockets

- Used for real-time communication
- Earlier used as `ws` and `wss`
- Helps send continuous updates to UI (like live weather data)

Day 4

Git and Source Control

Git helps teams work together without overwriting each other's code.
It stores all versions of the project like a timeline.

Basic Git Commands

- `git add .` → Add all changed files
- `git add file.py` → Add single file
- `git commit -m "message"` → Save a version
- `git push` → Upload code to remote repo
- `git pull` → Get updated code from repo
- `git branch` → View branches
- `git checkout -b newbranch` → Create new branch
- `git checkout branchname` → Switch branch
- `git stash` → Temporarily save changes
- `python -m venv venv` → Create virtual environment
- `venv/Script/activate` → Activate environment
- `pip install -r requirements.txt` → Install all libraries

Day 5

Databases & Normalization

Definitions

- **DB:** A place to store information
- **DBMS:** Software that manages the data
- **CRUD:** Create, Read, Update, Delete (basic database operations)

Types of Databases

- **Structured:** SQL, PostgreSQL
- **Unstructured:** MongoDB

Normalization (Simple)

Normalization means arranging data so that:

- There is no unnecessary repetition
- The database becomes efficient

Forms include:
1NF, 2NF, 3NF, BCNF, 4NF, 5NF

Dependencies

- **Partial dependency:** Column depends on part of composite key
- **Transitive dependency:** Column depends indirectly on another column

Keys

- **Primary Key:** Unique identifier
- **Foreign Key:** Connects two tables
- **Composite Key:** Two or more columns used together as a key

Web Pages to Develop

- Dashboard (graphs, AQI values)
- About page
- Report page (documents)
- Login / Admin panel

Day 6

AI Model Summary

Providers:

Google, OpenAI, Grok, Microsoft

Models:

Gemini, GPT, Llama, Copilot

Libraries:

openai, genai, grok

Terms:

- **LLM:** Large Language Model
- **Prompt Engineering:** Giving instructions to control model output

Machine Learning Basics

- **Supervised Learning:** Uses labeled data (Regression & Classification)
- **Unsupervised Learning:** Uses unlabeled data (Clustering, K-means)
- **Reinforcement Learning:** Learns by trial and error

Postgres Notes

- Used with pgAdmin
- Often used for vector storage
- **Embedding:** Converting text into numeric vectors
- **NLTK:** Library for text processing

Day 7

NLTK (Natural Language Toolkit)

NLTK is a Python library used for **text processing** and various NLP operations.

Key Features

- **Tokenization** – Splitting text into words or sentences
- **Stemming** – Reducing words to their root form
- **Lemmatization** – Reducing words to base form using vocabulary & grammar

Git Clone Steps (Full Guide)

Steps to clone a GitHub repository into your local system.

1. Install Git

Download and install Git from: https://git-scm.com/

2. Go to Your GitHub Repository

Open the repository you want to clone.

3. Copy HTTPS URL

Click on **Code** → Copy the **HTTPS URL**.

4. Open Command Prompt or Terminal

Use `cmd`, PowerShell, or Linux Terminal.

Day 8

Milestones, Documentation & Final Workflow

Milestones

- Milestone 1 → Thursday & Friday(25%)
- Milestone 2 → 27th & 28th Nov(50%)
- No major UI changes after 30 Nov(75%)
- Final demo on **1st December**

Git Hub Process

- Maintain both team repo and personal repo
- Add all teammates as contributors
- Keep UI transparent and clean
- Maintain backup copies
- Upload PPT, code, and documents

Final Submission

- A complete **45-page document**
- Must include all daily notes, concepts, screenshots, and project updates
- Completion percentage includes UI + backend + ML model + DB

Corporate Communication Tip

Speak openly and confidently like in a workplace environment.
Every class will introduce a new concept—document everything neatly.

Day -10

ANN – Artificial Neural Network

- Basic neural network with input, hidden, and output layers
- Works well for **simple structured data**
- Cannot handle sequence or image patterns well

RNN – Recurrent Neural Network

- Designed for **sequence data**
- Has feedback loops so previous output becomes next input
- Used for **text, speech, time series**
- Suffers from **vanishing gradient problem**

CNN – Convolutional Neural Network

- Detects patterns like edges, shapes, objects
- Uses **convolution → pooling → dense layers**

LSTM – Long Short-Term Memory

- Improved RNN that handles **long-term dependencies**
- Solves the **vanishing gradient** issue
- Has 3 gates:

2. LSTM Gates

1.Forget Gate

- Decides **what past information to delete**
- Uses a sigmoid (0 to 1 → delete to keep)

2.Input Gate

- Decides **what new information to store**
- Has sigmoid + tanh

3.Output Gate

- Decides **what to output as hidden state**
- Controls what next cell receives

Together:
LSTM remembers important info and forgets unnecessary info → best for NLP.

3. NLP (Natural Language Processing)

NLP makes machines understand human language.

Steps to apply NLP to any project:

4. Implementation Steps

1. Collect Data

- Text, sentences, reviews, tweets, emails, etc.

2. Prepare & Engineer Features

- Clean text
- Tokenize
- Remove stopwords
- Convert words to numbers (TF-IDF, Bag-of-words, embeddings)

3. Train a Model

- ML models → Logistic Regression, SVM
- DL models → RNN, LSTM, Transformers

4. Evaluate Using Metrics

- Accuracy
- Precision
- Recall
- F1 score
- Confusion matrix

Evaluation is done on holdout/test data.

DAY 11

NLTK – Key Steps

1.Tokenization

Breaking sentence into words or sentences.
Example: `"I love AI"` → `["I", "love", "AI"]`

2. Stop Words Removal

Removing common words that add no meaning.
Examples: *the, is, are, an, with*

3. Stemming

Cutting words to base form (may be not meaningful).
Example: "playing" → "play"

4. Lemmatization

Converts word to **dictionary form**.
Example: "better" → "good"

5. POS Tagging

Part of Speech tagging
Example: Noun, Verb, Adjective

6.NER

Named Entity Recognition
Find names, places, dates, organizations.

TF-IDF (Term Frequency – Inverse Document Frequency)

Used to convert text → numeric vectors.

Formula:

- **TF:** how many times a word appears
- **IDF:** how rare the word is across documents

Rare but important words get **higher weight**.

Used in:

- Classification
- Search engines
- Information retrieval

Logistic Regression

- A **classification algorithm**
- Uses **sigmoid function** to predict probability
- Best for **binary classification**

Formula:

```
sigmoid(z) = 1 / (1 + e^-z)
```

SVM (Support Vector Machine)

Mainly used for classification

SVM finds a **hyperplane** that separates classes.

Margin

Distance between the hyperplane and closest data points (support vectors).

Hard Margin SVM

- Perfect separation
- No misclassification allowed
- Very strict
- Works only when data is linearly separable

Soft Margin SVM

- Allows some misclassification
- More practical
- Works on real-world data
- Uses **C parameter** (penalty)

DAY 12

Reinforcement Learning (RL)

Reinforcement Learning is a type of AI where an **agent learns by doing actions** and receiving **rewards** or **punishments**.
The agent keeps trying, learns from mistakes, and improves.

More than one agent can work together (multi-agent).
Agents can work with or without a human.
Autogen is a tool used to build agent-based systems.

CSS Flexbox

Flexbox is used to **arrange items on a webpage** easily.

Two types of sizes:

- **Intrinsic size** → natural size of the element
- **Extrinsic size** → size given by the developer

Handling Overflow in CSS

If content is bigger than its box, we use **overflow**:

- visible → content shows outside
- hidden → extra content is cut
- scroll → scrollbars always
- auto → scrollbars only when needed

We can separately control horizontal and vertical overflow.

Box Sizing

Box sizing controls how element size is calculated.

Two types:

- **content-box** → width/height include only content
- **border-box** → padding and border included inside width/height

Layout Design

Two main layout methods:

- CSS Grid
- Flexbox

Flexbox uses:

- display: flex
- flex-direction

HTML & CSS Basics

Small introduction to HTML tags and basic CSS styling with examples.

Python Integration

Python works with frontend using:

- Flask
- FastAPI

These are used to create backend APIs.

Git vs GitHub

Git → version control software on your computer
GitHub → online website to store and share Git projects

DAY 13

Flex Wrap

Flex-wrap controls whether items:

- stay on one line
- move/wrap to next line
- wrap in reverse order

Flex Container Concepts

Any `<div>` can become a flex container.
All direct children become flex items.

Each item can align itself using:

- flex-start
- center
- flex-end
- stretch
- auto

Order Property

Order changes the placement of items **without changing HTML**.

- Lower order → appears earlier
- Higher order → appears later

Creating an OpenAI API Key

Steps in simple form:

- Search for "OpenAI API key"
- Go to API Keys page
- Create a new key
- Install OpenAI library
- Import it into Python

- Start using the client

Upcoming Tasks

Tasks planned for Thursday & Friday:

- Use API key
- Build chatbot
- Build Word document generator
- Push code to GitHub
- Prepare PPT for Milestone 2

Real-Time Data

To show real-time data:

- Database must be connected to the UI
- UI should fetch updated values automatically

Python Chat Completion Example

Simple code:

```
resp = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": "Write a two-line poem about chai."}
        ]
)
```

Meaning of Key Terms

- model → which AI model you use
- messages → conversation history
- system → rules for assistant
- user → your question
- max_tokens → how long the answer can be
- temperature → creativity level

DAY-14

Humanised Notes

Random Forest

Random Forest is a machine learning technique that builds **many decision trees** instead of just one.
Each tree looks at a **random subset** of the data and features, so every tree learns something slightly different.

How it predicts:

- For **classification** → It takes the **majority vote** from all trees.
- For **regression** → It takes the **average output** of the trees.

Because it uses **multiple trees together**, it is called an **ensemble model**.

Code Example:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
```

Height Classification Example

Suppose you have the heights of players:

- A = 170 cm, B = 168 cm → **Fit for sports**
- C = 150 cm, D = 152 cm → **Not fit for sports**

If you get new heights like **169** or **151**, you can use a classification model (like Random Forest or Decision Tree) to predict whether they are fit for sports.
This is a simple example showing **how classification works in ML**.

K-Means Clustering

K-Means is an **unsupervised learning algorithm**, meaning it groups data **without using labels**.

What it does:
It divides data into **K clusters** based on similarity.

Steps:

1. Pick **K** initial centroids.
2. Assign each data point to the **nearest centroid**.
3. Recalculate each centroid based on the mean of its cluster.
4. Repeat until clusters don't change.

Objective:
K-Means tries to reduce the total distance between points and their centroids.

Code Example:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
```

14.7 Linear Regression

Linear Regression is used for **predicting continuous values** such as price, temperature, sales, etc.

Equation:

y=mx+cy = mx + cy=mx+c

Where:

- **m** = slope
- **c** = intercept

Mean Squared Error (MSE)

MSE is a **loss function** used in regression.

It calculates how far the predicted value is from the actual value.

Formula:

MSE=1n∑(y−y^)2MSE = \frac{1}{n} \sum (y - \hat{y})^2MSE=n1∑(y−y^)2

Linear Regression Code

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

XGBoost

XGBoost is a very powerful and efficient **gradient boosting algorithm**.

It builds trees **one after another** (sequentially).
Every new tree tries to **correct the errors** made by the previous trees.

Update formula:

Fnew(x)=Fold(x)+η·h(x)F_{\text{new}}(x) = F_{\text{old}}(x) + \eta \cdot h(x)Fnew(x)=Fold(x)+η·h(x)

Where:

- **η (eta)** = learning rate
- **h(x)** = a small tree (weak learner)

Key Features:

- Sequential tree building
- Error correction at every step
- Regularization to avoid overfitting

Code Example:

```
from xgboost import XGBClassifier
model = XGBClassifier()
```

Day 15 – MySQL

What is MySQL?

MySQL is a database system that uses **SQL** (Structured Query Language) to store, manage, and retrieve data.

Example:

```
SELECT * FROM table_name;
```

Common SQL Commands

- **SELECT** – retrieve data
- **INSERT** – add new data
- **UPDATE** – change existing data
- **DELETE** – remove data
- **CREATE** – create tables/databases
- **ALTER** – modify a table
- **DROP** – delete a table/database
- **RENAME** – rename a table
- **TRUNCATE** – delete all rows from a table

WHERE Clause

Used to filter rows based on conditions.

Example:

```
SELECT * FROM Customer WHERE country = 'Mexico';
```

Operators

- **Comparison:** =, >, <, >=, <=, !=
- **BETWEEN:** range check
- **LIKE:** pattern matching
- **IN / NOT IN:** set-based filtering
- **AND, OR, NOT:** logical operators
- **ORDER BY:** sorting

LIKE Pattern Matching Examples

- **a%** → starts with a
- **%a** → ends with a
- **%a%** → contains a

- **_a%** → second letter is a
- **a_%** → starts with a, minimum 2 characters
- **a__%** → starts with a, minimum 3 characters
- **a%o** → starts with a and ends with o

Aggregate Functions

- **MIN()** – smallest value
- **MAX()** – largest value
- **COUNT()** – number of rows
- **AVG()** – average value

Aliases

Used to rename a column or table temporarily.

Example:

```
SELECT name AS student_name FROM student;
```

Joins

- **INNER JOIN** – matching rows only
- **LEFT JOIN** – all left table rows + matches
- **RIGHT JOIN** – all right table rows + matches
- **FULL JOIN** – all records from both tables
- **CROSS JOIN** – Cartesian product

Example tables:

- student (std_id, std_name, address)
- subject (sbj_id, stud_id, sbj_name)

UNION & UNION ALL

- **UNION** – merges results and removes duplicates
- **UNION ALL** – keeps duplicates

GROUP BY & HAVING

- **GROUP BY** – groups rows by column
- **HAVING** – applies conditions on grouped data (used with aggregates)

Day 16

Today was mainly focused on preparing for **Milestone–2**. I revised the project requirements, checked previous work, and planned what needed to be completed.

Day 17

Worked actively on **Milestone–2** tasks. I implemented the planned changes and coordinated with my team to ensure everything was progressing correctly.

Day 18

Continued and completed the remaining **Milestone–2 work**. I tested the outputs and made sure the milestone goals were achieved.

Day 19

Today, our team **presented the UI** that we had designed.
Along with that, I learned and practiced **Python built-in string methods**, which are very useful for text processing.

I learned how to:

- python build in methods:

1. lower():Converts the string to lowercase
2. upper():Converts the string to uppercase
3. split():Splits the string into a list (default: by spaces).
4. strip():Removes leading and trailing spaces (or given characters).
5. join():Joins elements of a list into a string using a separator.
6. replace():Replaces part of a string with another string.
7. startswith():Checks if the string starts with a given value → returns True/False.
8. endswith():Checks if the string ends with a given value → returns True/False.
9. find():Returns the index of a substring (or -1 if not found).
10. isdigit():Returns True if all characters are digits.

Day 20

LIST METHODS

- **append()**
  Adds one element to the end of the list.
- **extend()**
  Adds multiple elements to the end of the list.
- **insert()**
  Adds an element at a specified position.
- **remove()**
  Removes the first matching element from the list.
- **pop()**
  Removes and returns an element from the list.
- **index()**
  Returns the position of a specified element.
- **count()**
  Returns the number of times an element appears.
- **sort()**
  Sorts the list in ascending order.
- **reverse()**
  Reverses the order of elements in the list.

DICTIONARY METHODS

- **get()**
  Returns the value of a key safely.
- **keys()**
  Returns all keys in the dictionary.
- **values()**
  Returns all values in the dictionary.
- **items()**
  Returns key–value pairs.
- **update()**
  Adds or updates key–value pairs.
- **pop()**
  Removes and returns the value of a specified key.
- **popitem()**
  Removes and returns the last inserted key–value pair.
- **clear()**
  Removes all items from the dictionary.

SET METHODS

- **add()**
  Adds an element to the set.
- **remove()**
  Removes a specified element from the set.
- **discard()**
  Removes an element without causing an error.
- **pop()**
  Removes a random element from the set.
- **clear()**
  Removes all elements from the set.
- **union()**
  Returns all unique elements from two sets.
- **intersection()**
  Returns common elements between sets.
- **difference()**
  Returns elements present in one set but not in another.

FILE I/O METHODS

- **open()**
  Opens a file.
- **read()**
  Reads the entire content of a file.
- **readline()**
  Reads one line from a file.
- **readlines()**
  Reads all lines and returns them as a list.
- **write()**
  Writes data to a file.
- **writelines()**
  Writes multiple lines to a file.
- **close()**
  Closes the file.

GENERAL PURPOSE FUNCTIONS

- **len()**
  Returns the length of an object.
- **range()**
  Generates a sequence of numbers.
- **print()**
  Displays output on the screen.
- **type()**
  Returns the data type of a value.

- **id()**
  Returns the memory identity of an object.
- **sorted()**
  Returns a sorted version of an iterable.
- **enumerate()**
  Returns index and value pairs.
- **zip()**
  Combines elements from multiple iterables.

Day 21

This day was about learning **advanced Python concepts**.

- **int()**
  Converts a value into an integer.
- **float()**
  Converts a value into a floating-point number.
- **str()**
  Converts any value into a string.
- **list()**
  Converts an iterable into a list.
- **tuple()**
  Converts a collection into a tuple.
- **set()**
  Converts a collection into a set and removes duplicate values.
- **dict()**
  Creates a dictionary using key–value pairs.

Mathematical Functions

- **abs()**
  Returns the positive value of a number.
- **sum()**
  Returns the total of all elements in an iterable.
- **min()**
  Returns the smallest value.
- **max()**
  Returns the largest value.
- **pow()**
  Returns a number raised to a power.
- **round()**
  Rounds a number to the specified decimal places.

Functional Programming Tools

- **filter()**
  Returns elements that satisfy a condition.
- **map()**
  Applies a function to every element in an iterable.
- **reduce()**
  Reduces an iterable to a single value.
- **Lambda function**
  A small anonymous function written in one line.

Input and Output

- **input()**
  Takes input from the user as a string.
- **format()**
  Formats values into a string.

Class and Object Related Methods

- **getattr()**
  Retrieves the value of an object attribute.
- **setattr()**
  Sets or updates an object attribute.
- **hasattr()**
  Checks whether an object has a specific attribute.
- **delattr()**
  Deletes an attribute from an object.
- **isinstance()**
  Checks whether an object belongs to a specific class.
- **issubclass()**
  Checks whether a class is derived from another class.

Day 22

Python Data Types

Python data types define the kind of value a variable can store, such as numbers, text, collections, or logical values.

PEP 8

PEP 8 is the official coding style guide of Python that helps programmers write clean, readable, and consistent code.

Mutable and Immutable Objects

Mutable objects can be changed after they are created.
Immutable objects cannot be changed once they are created.

Built-in Features of Python

Python includes built-in functions, a large standard library, error handling support, dynamic typing, and strong community support.

Python Memory Management

Python automatically manages memory using reference counting and garbage collection.

Indentation

Python uses indentation to define blocks of code instead of braces, which improves readability.

Python as an Interpreted Language

Python code is converted into bytecode and executed by the Python Virtual Machine.

Namespaces

A namespace is a space where variable names are stored to avoid name conflicts.

List and Tuple Difference

Lists are mutable and can be changed.
Tuples are immutable and cannot be modified.

Sets

Sets store unique and unordered elements and do not allow duplicates.

Dictionaries

Dictionaries store data in key–value pairs and allow fast access to values.

Merging Dictionaries

Merging dictionaries means combining two dictionaries into one.

Removing Duplicates from a List

Duplicates can be removed by converting the list into a set or by filtering elements.

Flattening a Nested List

Flattening means converting a list with inner lists into a single-level list.

Shallow Copy and Deep Copy

Shallow copy duplicates references to objects.
Deep copy duplicates the actual objects completely.

List Slicing

List slicing is used to extract a portion of a list.

Reversing a List

Reversing a list means changing the order of elements from last to first.

Frozen Set

A frozen set is an immutable version of a set.

Difference Between `is` and `==`

`==` checks whether values are equal.
`is` checks whether both variables refer to the same object.

Stack

A stack follows the Last In, First Out principle.

Queue

A queue follows the First In, First Out principle.

Day 23

Difference Between Functions and Methods

Functions are independent blocks of code, while methods are functions defined inside a class and are accessed using objects.

args and kwargs

args is used to accept a variable number of positional arguments, and kwargs is used to accept a variable number of keyword arguments.

Recursion

Recursion is a process in which a function calls itself to solve a problem.

Decorators

Decorators are functions that modify the behavior of other functions without changing their code.

Generators

Generators are functions that return values one at a time using the yield keyword, saving memory.

Iterator vs Generator

Iterators use explicit methods to fetch values, while generators automatically handle iteration using yield.

Static Method and Class Method

Static methods do not use class or object data, while class methods receive the class as their first argument.

Magic Methods

Magic methods are special methods with double underscores that define object behavior.

Monkey Patching

Monkey patching is the process of modifying methods or functions at runtime.

Error vs Exception

Errors are serious problems that stop program execution, while exceptions are runtime issues that can be handled.

Custom Exception

A custom exception is a user-defined exception created by extending the Exception class.

Exception Chaining

Exception chaining links one exception to another to show the original cause of an error.

Module vs Package

A module is a single Python file, while a package is a collection of modules in a directory.

Python Virtual Environment

A virtual environment is used to isolate project-specific dependencies.

tell() vs seek()

tell() returns the current file pointer position, while seek() changes the file pointer position.

os.path.exists()

Checks whether a file or directory exists.

Absolute vs Relative Path

An absolute path gives the complete location from the root, while a relative path is based on the current directory.

Pivot in Pandas

Pivot reshapes data by converting column values into headers.

GroupBy in Pandas

GroupBy groups data based on column values and applies aggregation operations.

Concat vs Append

concat() combines multiple datasets, while append() adds rows and is deprecated.

Global Interpreter Lock

GIL allows only one thread to execute Python bytecode at a time.

Metaclass

A metaclass is a class that defines how other classes are created.

Multithreading

Multithreading allows multiple threads to run concurrently, mainly for I/O tasks.

Multiprocessing

Multiprocessing uses multiple processes to execute tasks in parallel, bypassing the GIL.

Feature Scaling

Feature scaling adjusts numerical values to improve machine learning model performance.

Train and Test Split

Splitting divides data into training and testing sets.

Features of Java

Java provides platform independence, object-oriented features, security, robustness, and automatic memory management.

JVM, JRE, and JDK

JVM executes bytecode, JRE provides runtime environment, and JDK includes development tools.

Access Modifiers

Access modifiers control the visibility of classes and class members.

== vs .equals()

== checks reference equality, while .equals() checks value equality.

Constructor

A constructor is a special method used to initialize objects.

Private Constructor

A private constructor restricts object creation from outside the class.

Overloading vs Overriding

Overloading uses the same method name with different parameters, while overriding redefines a method in a subclass.

Primitive Data Types

Primitive data types store basic values directly.

Type Casting

Type casting converts one data type into another.

Static vs Non-static Variables

Static variables belong to the class, while non-static variables belong to objects.

final vs finally vs finalize

final prevents modification, finally always executes in exception handling, and finalize is called before object destruction.

this and super

this refers to the current object, while super refers to the parent class.

String vs StringBuffer vs StringBuilder

String is immutable, StringBuffer is mutable and thread-safe, and StringBuilder is mutable and faster.

Array vs ArrayList

Arrays have fixed size, while ArrayLists are dynamic.

Collections Framework

The Collections Framework provides interfaces and classes for data storage and manipulation.

List vs Set

List maintains order and allows duplicates, while Set does not allow duplicates.

ArrayList vs LinkedList

ArrayList is faster for access, while LinkedList is faster for insertion and deletion.

Concurrent HashMap

Concurrent HashMap is a thread-safe map that allows concurrent access.

Fail-fast vs Fail-safe Iterators

Fail-fast iterators throw errors on modification, while fail-safe iterators work on copies.

Multiple Catch Blocks

Multiple catch blocks handle different types of exceptions.

throw vs throws

throw explicitly throws an exception, while throws declares exceptions.

Process vs Thread

A process has its own memory space, while a thread shares memory within a process.

Synchronization

Synchronization controls access to shared resources in multithreading.

sleep() vs wait()

sleep() pauses execution without releasing locks, while wait() releases locks and waits.

Deadlock

Deadlock occurs when threads wait indefinitely for each other.

Stream API

Stream API supports functional-style operations on collections.

Serialization

Serialization converts an object into a byte stream.

Inner Class

An inner class is a class defined inside another class.

Compile-time vs Runtime Polymorphism

Compile-time polymorphism uses overloading, while runtime polymorphism uses overriding.

Method Hiding

Method hiding occurs when a static method in a subclass has the same name as in the parent class.

Day 24

Spent the day preparing for **Milestone–3** and working on project-related improvements.

Day 25

Software Development Life Cycle (SDLC)

SDLC is a step-by-step process followed to design, develop, test, deploy, and maintain software in a structured way.

SDLC Roles

- **Functional Consultant / Business Analyst**
  Understands business needs and converts them into software requirements.
- **Developer / Programmer**
  Writes and implements the code based on requirements.
- **Testing / QA Team**
  Tests the software to find bugs and ensure quality.
- **Administration Team**
  Manages servers, databases, and system infrastructure.
- **Product / Project Management**
  Plans, monitors, and controls the project timeline and delivery.
- **Sales and Marketing**
  Promotes and sells the software product.
- **Human Resource**
  Manages hiring, employee roles, and team coordination.
- **Documentation Team**
  Prepares manuals, guides, and technical documents.
- **Support Team**
  Provides help and maintenance after deployment.

Documents in SDLC

- **Functional Specification Document (FSD)**
  Describes what the system should do from a business perspective.
- **Technical Specification Document (TSD)**
  Describes how the system will be built technically.

- **Test Script**
  Contains steps to test software functionality.
- **Test Result**
  Records outcomes of testing activities.

Development Process

- Requirement gathering
- Prototype creation
- Software development
- Unit testing
- Quality testing
- Production deployment

This process ensures software is built correctly and delivered reliably.

Servers

- **Development Server**
  Used by developers to write and test code.
- **Quality (QA) Server**
  Used for testing the application.
- **Production Server**
  Used by real users in live environment.

Testing / QA

Testing is done based on requirements documents, mainly on the QA server, and test results are recorded to ensure software quality.

Administration

Administration handles system operations such as network setup, database maintenance, and application availability.

Waterfall Model

Waterfall is a linear development model where each phase must be completed before moving to the next phase.
It is suitable for small projects with fixed requirements but not flexible to changes.

Agile Methodology

Agile is an iterative and flexible development model where work is divided into small parts called sprints.
It supports frequent feedback, continuous testing, and quick delivery.

Agile Roles

- **Product Owner**
  Defines product requirements and priorities.
- **Scrum Master**
  Facilitates Agile practices and removes obstacles.
- **Development Team**
  Designs, develops, and tests the product.

Agile Practices

- Daily stand-up meetings
- Sprint planning
- Sprint review
- Sprint retrospective

These practices improve communication and productivity.

DAY 26 – Milestone Preparation

Project Preparation for Milestone Three

This phase focuses on reviewing progress, refining features, fixing issues, and ensuring readiness for evaluation.

DAY 27 – Algorithms and Sorting Techniques

*A Algorithm**

A* is a path-finding algorithm used to find the shortest and most efficient path between two points using cost and heuristic values.

Bubble Sort

Bubble Sort repeatedly compares adjacent elements and swaps them until the list becomes sorted.

Selection Sort

Selection Sort finds the smallest element from the unsorted part and places it at the correct position.

Insertion Sort

Insertion Sort builds the sorted list gradually by inserting elements into their correct position.

Merge Sort

Merge Sort divides the array into smaller parts, sorts them, and merges them back into a sorted array.

Quick Sort

Quick Sort selects a pivot element and arranges elements around it using recursion.

Heap Sort

Heap Sort uses a heap data structure to sort elements efficiently.

DAY 28 – Milestone Three

Milestone Three

Milestone Three represents a major evaluation stage where core features are implemented and tested.

DAY 29 – Milestone Three

Milestone Three Explanation

This stage focuses on finalizing functionality, improving performance, validating requirements, fixing bugs, and preparing the project for final submission or deployment.

DAY-30

Q-Learning Algorithm

Q-Learning is a **reinforcement learning algorithm** where an agent learns the best action to take in each state by **trial and error**, using rewards from the environment. It does **not need a model** of the environment.

Main Components

1. **Agent**
   The learner/decision maker (e.g., parking controller).
2. **Environment**
   Everything the agent interacts with (e.g., parking lot).
3. **State (s)**
   Current situation of the environment.
4. **Action (a)**
   Decision taken by the agent.
5. **Reward (r)**
   Feedback after taking an action.
6. **Q-value Q(s, a)**
   Expected future reward for taking action `a` in state `s`.

Meaning of Terms

- **α (alpha)** → Learning rate (how fast learning happens)
- **γ (gamma)** → Discount factor (importance of future rewards)
- **r** → Reward received
- **s′** → Next state
- **max Q(s′,a′)** → Best future action value

Q-Learning Algorithm (Step-by-Step)

1. Initialize the **Q-table** with zeros
2. Observe the current **state (s)**
3. Choose an **action (a)**
   - Explore (random) or exploit (best known)
4. Perform the action
5. Receive **reward (r)** and **next state (s′)**
6. Update Q(s, a) using the formula
7. Set **s = s′**
8. Repeat until goal is reached
9. Repeat for many episodes until learning converges

DAY-31

Deep Learning is a subset of **Machine Learning** that uses **neural networks with many layers (deep neural networks)** to automatically learn patterns from large amounts of data.

How Deep Learning Works

1. **Input Layer** – Receives raw data (image pixels, text, numbers)
2. **Hidden Layers** – Multiple layers extract features step by step
    - Early layers → simple features
    - Deeper layers → complex features
3. **Output Layer** – Gives final prediction or classification

Key Components

- **Neurons** – Basic processing units
- **Weights & Bias** – Learnable parameters
- **Activation Functions** – ReLU, Sigmoid, Softmax
- **Loss Function** – Measures error
- **Backpropagation** – Adjusts weights
- **Optimizer** – Adam, SGD, RMSprop

1 Neural Network Overview

A **neural network** consists of:

- **Input layer**
- **One or more hidden layers**
- **Output layer**

Each layer has:

- **Weights (w)**
- **Bias (b)**
- **Activation function (f)**

The training of a neural network happens in **two phases**:

1. **Forward Pass**
2. **Backpropagation**

1. Forward Pass

The **forward pass** is the process in which input data is propagated through the neural network from the **input layer to the output layer** to generate a predicted output and compute the loss.

In the forward pass, each neuron receives input values, multiplies them by corresponding weights, adds a bias, and applies an activation function. This transformation is performed layer by layer until the final output is obtained.

The forward pass **does not modify weights**. Its role is strictly to **evaluate the current model performance** by producing predictions and measuring error using a loss function.

Information Flow

Input Layer → Hidden Layers → Output Layer

Main Operations in Forward Pass

1. Linear transformation (weighted sum)
2. Application of activation functions
3. Generation of predicted output
4. Computation of loss

Purpose of Forward Pass

- To generate predictions
- To measure how far predictions are from actual values
- To provide error information for learning

Characteristics

- Direction: Forward
- Learning: No
- Output: Prediction and loss
- Repeated for every training example

2. Backward Pass (Backpropagation)

The **backward pass**, also known as **backpropagation**, is the process of propagating the error backward through the network to compute gradients of the loss function with respect to weights and biases, enabling parameter updates.

During the backward pass, the error computed in the forward pass is traced backward from the output layer to the input layer. Using the **chain rule of calculus**, the network determines how much each weight and bias contributed to the error.

These gradients are then used by an optimization algorithm (such as gradient descent) to update parameters in a way that minimizes the loss.

Information Flow

Output Layer → Hidden Layers → Input Layer

Main Operations in Backward Pass

1. Error gradient calculation
2. Gradient computation for each layer
3. Weight and bias updates
4. Loss minimization

Purpose of Backward Pass

- To train the network
- To reduce prediction error
- To improve model accuracy over time

DAY-32

1. Depth First Search (DFS)

DFS explores a graph or tree by going **as deep as possible** along one branch before backtracking.

Working Principle

- Start from a node
- Visit one adjacent node
- Keep moving deeper
- If no unvisited neighbor exists → **backtrack**

Data Structure Used

- **Stack** (explicit stack or recursion)

Algorithm (Steps)

1. Push the start node into the stack
2. Mark it as visited
3. While stack is not empty:
    - Pop a node
    - Visit all unvisited adjacent nodes
    - Push them into the stack

Example

Graph: `A → B → D`, `A → C → E`
DFS traversal: **A → B → D → C → E**

Applications

- Maze solving
- Cycle detection
- Topological sorting
- Path finding

2. Breadth First Search (BFS)

Definition

BFS explores all neighboring nodes **level by level** before moving deeper.

Working Principle

- Start from a node
- Visit all its neighbors
- Then visit neighbors of neighbors

Data Structure Used

- **Queue**

Algorithm (Steps)

1. Enqueue the start node
2. Mark it as visited
3. While queue is not empty:
    - Dequeue a node
    - Visit all unvisited adjacent nodes
    - Enqueue them

Example

Graph: `A → B, C, B → D, C → E`
BFS traversal: **A → B → C → D → E**

Depth Limited Search (DLS)

Depth Limited Search is a **modified version of DFS** where the search is allowed to go **only up to a fixed depth limit**.

Why DLS is needed

In DFS:

- The search may go **infinitely deep**
- Especially in **cyclic or infinite graphs**

DLS **prevents infinite loops** by restricting depth.

How it works (Explain step-by-step)

1. Start from the root node
2. Explore nodes depth-wise like DFS
3. Stop expanding a node if:
    - Depth limit is reached
4. If goal is found → success
5. If limit reached without goal → cutoff

Uniform Cost Search (UCS)

Uniform Cost Search expands the node with the **lowest path cost from the start node**.

Key Idea (Important line)

Cost matters, not depth

How UCS works

1. Start from initial node
2. Store nodes in a **priority queue**
3. Expand node with **minimum cumulative cost**
4. Repeat until goal is reached

Bidirectional Search

Bidirectional Search performs **two searches simultaneously**:

- One from the **start**
- One from the **goal**

The search stops when both meet.

Instead of searching one way, search **from both ends**

How it works

1. Start BFS from start node
2. Start BFS from goal node
3. Expand both searches level by level
4. Stop when a common node is found

Beam Search

Definition

Beam Search is a **heuristic search algorithm** that expands only the **best `k` nodes** at each level.

Key Term

Beam Width (k)

- Number of best nodes kept

How it works

1. Start from root
2. Expand all children
3. Evaluate nodes using heuristic
4. Keep only top `k` nodes
5. Discard others
6. Repeat until goal

DAY-33

Worked on the project and identified and fixed existing bugs to improve the stability and performance of the application.

DAY-34

Integrated additional features/components into the project, performed testing to ensure proper functionality, and provided feedback to the mentor regarding the class sessions.

DAY-35

Milestone Submission (Implementation Phase)

Demonstrated the successful implementation of the core functionalities of the project.

DAY-36

Milestone Submission (Testing, Documentation & Finalization)

- The project objectives were successfully achieved.
- Testing and documentation were completed and submitted.