A PROJECT REPORT

ON

STATE CAST-DYNAMIC WEATHER VISUALIZATION

Submitted in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

Submitted By

D. Sadhana	**(2071085)**
D. Shahara Banu	**(2071094)**
M. Sharada	**(2071096)**

Under the guidance of

Dr . A. Supriya, MTech, Ph.D.
Assistant Professor(Senior)

Accredited by NAAC with "A+" Grade
ISO 21001:2018

SCHOOL OF ENGINEERING AND TECHNOLOGY
SRI PADMAVATI MAHILA VISVAVIDYALAYAM
(Women's University)
Tirupati
APRIL 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOLOF ENGINEERING AND TECHNOLOGY

SRI PADMAVATI MAHILA VISVAVIDYALAYAM

(WOMEN'S UNIVERSITY)

TIRUPATI – 517502, ANDHRA PRADESH

Accredited by NAAC with 'A+' Grade

DECLARATION BY THE CANDIDATE

We hereby declare that the project entitled "**State Cast-Dynamic Weather Visualization**" submitted by us to the Department of Computer Science and Engineering, School of Engineering and Technology, Sri Padmavati Mahila Visvavidyalayam, Tirupati in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by us under the supervision of **Dr. A. Supriya, Assistant Professor**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this university or of any other institute or university.

Place: Tirupati

Date:

Signature of the Student

D.Sadhana

D.Shahara Banu

M.Sharada

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOLOF ENGINEERING AND TECHNOLOGY

SRI PADMAVATI MAHILA VISVAVIDYALAYAM

(WOMEN'S UNIVERSITY)

TIRUPATI – 517502, ANDHRA PRADESH

Accredited by NAAC with 'A+' Grade

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**State Cast-Dynamic Weather Visualization**" submitted by **D. Sadhana(2071085), D. Shahara Banu(2071094), M.Sharada(2071096)** to School of Engineering and Technology, Sri Padmavati Mahila Visvavidyalayam, Tirupati in partial fulfillment of the requirement for the award of the degree of B. Tech in Computer Science and Engineering is a record of bonafide work carried out by them under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Guide:

Dr. A.Supriya ,M.Tech, Ph.D.,

Assistant Professor (Senior)

Department of CSE

Prof. V. Saritha

Head of the Department,

SOET, SPMVV.

Internal Examiner

External Examiner

CONTENTS

ACKNOWLEDGEMENT

Our sincere appreciation and thanks to all those who have helped me to complete this long-term undertaking successfully. A long-term endeavour can only succeed with the support and guidance of many well-wishers.

It is very auspicious moment, and we would like to express my gratitude to beloved Vice Chancellor **Prof. D. BHARATHI** and Registrar **Prof. N. RAJINI** for providing a wonderful academic to complete this project.

I derive great pleasure in expressing our sincere gratitude to our Director **Prof. P. MALLIKARJUNA**, School of Engineering and Technology, SPMVV for providing better infrastructural facilities, which helped us to complete the project work successfully.

It is an auspicious moment that we would like to express our gratitude to **Prof. V. SARITHA**, Head of the Department, CSE for her consistent encouragement during the progress of this project.

I sincere thanks to our Project Coordinator **E. RAMESH BABU**, Assistant Professor, CSE and Engineering for his continuous help and support.

I sincerely thank to our project guide **Dr. A.SUPRIYA**, Assistant Professor, CSE for the ideas that led to complete the project work. I also thank her for her continuous guidance, support, and unfailing patience, throughout the coursework despite her busy schedule.

My sincere thanks to **Dr. JYOTI NARAYAN BHATE** Scientific Officer 'D' for his patience and consistent encouragement and mentoring us during 4 months internship at National Atmospheric Research Laboratory.

I sincerely thank to teaching and non-teaching staff of CSE Department for their help and support. We derive a great pleasure in expressing our sincere gratitude to our family and friends for their love and support.

ABSTRACT

This project aims to develop a comprehensive weather forecast application using full stack development technologies. The application will provide users with real-time weather updates, forecasts, and relevant one week weather data of a particular selected state using an intuitive user interface. The weather data contains surface temperature, rainfall, humidity and wind speed of a particular selected state and present it to users up to one week from the selected date using a scroll bar at the bottom through a web page. Overall, this weather forecast application will offer users a seamless experience to access accurate weather information anytime, anywhere.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 General Introduction

Weather forecast plays a crucial role in our lives, influencing daily activities, travel plans, agriculture, and various industries. It provides valuable information about the atmospheric conditions expected in a specific location over a certain period, typically ranging from a few hours to several days. This essay explores the significance of weather forecasting, its methods, and the technology involved, along with its impact on society.Weather forecasting is the process of predicting the state of the atmosphere at a specific location and time. It involves collecting data from various sources such as satellites, radar, and weather stations, and using this data to create models that simulate the behaviour of the atmosphere. These models help meteorologists make predictions about future weather conditions, including temperature, precipitation, wind speed, and cloud cover.

Meteorologists use a variety of instruments, such as weather stations, satellites, and radar systems, to collect data on temperature, humidity, air pressure, and wind speed and direction.Weather forecasts are important for planning daily activities, agriculture, aviation, and emergency response, among other applications.Humans want weather forecasts to be accurate, timely, and easily accessible. They rely on forecasts to plan their daily activities, such as what to wear, whether to travel, or whether to engage in outdoor events.People prefer forecasts that are easy to understand, using simple language and visual representations like charts or graphs. They also appreciate forecasts that provide information about various weather parameters such as temperature, precipitation, wind speed, and humidity.Timeliness is crucial, as people want to know the weather conditions for the current day as well as the coming days. Forecasts that are too far in advance may be less useful due to their lower accuracy.

Importance of Weather Forecast:

- ❖ Safety and Preparedness: Weather forecasts help individuals and communities prepare for extreme weather events such as hurricanes, tornadoes, and blizzards. Advance warning allows people to take necessary precautions and evacuate if necessary, reducing the risk of injury and loss of life.
- ❖ Agriculture: Farmers rely on weather forecasts to plan their planting, irrigation, and harvesting activities. Accurate forecasts help them make informed decisions about crop selection and management practices, ultimately affecting crop yields and food production.
- ❖ Transportation: Weather forecasts are crucial for air, land, and sea transportation. Airlines use forecasts to plan flight routes, while road and rail operators use them to anticipate and mitigate the impact of adverse weather conditions on travel.
- ❖ Energy and Utilities: Energy companies use weather forecasts to predict energy demand and plan the production and distribution of electricity and gas. Forecasts also help utilities prepare for and respond to severe weather events that may disrupt services.
- ❖ Tourism and Recreation: Weather forecasts influence travel decisions, especially for outdoor activities such as hiking, camping, and skiing. Tourist destinations rely on accurate forecasts to manage visitor flows and ensure visitor safety.

- ❖ Overall, Humans want weather forecasts that are accurate, easy to understand, and available in a timely manner.

1.2 Problem Statement

Making prediction on weather trends is necessary but not so easy. This can be made easy only by taking the reference from the previous data.

Existing System

- The Existing applications cannot predict the weather short term efficiently. They used only small limited areas for weather forecasting. Accurate weather prediction is a difficult task due to dynamic change of atmosphere. It is susceptible for predicting weather in large areas at a time.
- The existing system provide weather forecast to the selected date, not upto one week. It displays the weather forecast parameters in the single graph which leads to confuse to analyse.

Disadvantages

- Existing applications struggle to predict short-term weather efficiently. This could be due to a lack of real-time data integration, sophisticated prediction models, or computational resources.
- Weather forecasting systems often focus on limited areas, which can be problematic for regions where weather conditions change rapidly over vast areas, like large countries or continents.
- The current systems might not provide weather forecasts for up to a week, limiting their usefulness for long-term planning.
- Presenting weather forecast parameters in a single graph can be confusing and may not provide a clear understanding of the predicted weather conditions.

1.3 Objective of the Proposed System

Users can search for weather forecasts state wise with particular date, with support for global coverage. The application allows users to access to historical weather data to track past weather patterns and trends for analysis and planning purposes.The application is designed to be accessible across multiple devices, including desktops, tablets, and smartphones, ensuring users can access weather information anytime, anywhere. The application displays weather forecast parameters from selected date to one week for a selected state through dropdown on the same web page.

Advantages

- Allowing users to access historical weather data for analysis and planning purposes enables them to track past weather patterns and trends, which can be valuable for various industries, including agriculture, transportation, and tourism.

- Being accessible across multiple devices, including desktops, tablets, and smartphones, enhances the usability and convenience of your application, ensuring that users can access weather information anytime, anywhere.
- The application's interface, which displays weather forecast parameters from the selected date to one week for a selected state through a dropdown on the same web page, provides a user-friendly experience that is easy to navigate and understand, enhancing user satisfaction.
- By providing accurate and up-to-date weather information, your application helps users make better decisions related to outdoor activities, travel plans, event scheduling, and more, ultimately enhancing their overall experience and safety
- By providing weather forecasts state-wise with support for global coverage, your application offers users a comprehensive view of weather conditions, helping them make informed decisions regardless of their location

1.4 Report Organization

Project Title: “State Cast-Dynamic Weather Visualization”

Objective:

State cast dynamic weather visualisation stores the previous weather data in the image format which helps to make future predictions. It helps the users to access the data up to 7 days from the start date. Through thus interactive application the user can view the weather data for any parameter and for a specific time using the movable time slider.

Project Guide: Dr. A. Supriya

Submitted by: D. Sadhana

D. Shahara Banu

M. Sharada

Submitted to: NATIONAL ATMOSPHERIC RESEARCH LABORATORY, GADANKI

CHAPTER 2
LITERARTURE SURVEY

2.1 Introduction

S NO	AUTHOR	TITLE	YEAR	METHODOLOGY	FEATURES	LIMITATIONS
1	Muthulakshmi A, Dr. S Baghavathi Priya [1]	“Automatic Monitoring and Controlling of Weather Condition using Big Data Analytics”.	2018	Big data Analytics	It gives efficient and accurate weather forecasting models to predict and monitor the weather datasets to predict rainfall.	It only predicts the rainfall for past data not for future data.
2	Smith et al., [2]	"Advancements in Location-Based Weather Forecasting"	2019	Machine Learning (ML) algorithms	Real-time data integration, high accuracy	Dependency on large datasets, model complexity
3	Brown & Jones[3]	“Image mapping for weather forecasting”	2020	Neural Networks (CNN) for image Analysis	High-resolution weather mapping, automated feature extraction	Limited interpretability, domain specificity
4	Wang et al.,[4]	“Deep Learning Techniques for Location-based Weather Prediction.”	2020	Deep Learning	Adaptability to complex patterns of weather prediction	Requirement of large datasets
5	Johnson & Lee[5]	“Relationship modelling for Location-based Weather	2021	Deep Learning(DL) models	Spatial-temporal relationships modelling	Spatial-temporal relationships modelling

		Forecasting”				
6	Liu & Gupta[6]	“Probabilistic Graphical Models for Location-based Weather Forecasting”	2021	Probabilistic graphical models	Probabilistic forecasts, uncertainty estimation	Limited scalability, parameter tuning issues
7	Patel and Sharma[7]	“Deep Learning Approaches for Location-Based Weather Forecasting”	2021	Deep Learning	Feature extraction from satellite imagery for improved forecasts	Dependence on data availability and quality
8	David Clark, Rachel White[8]	“Enhancing Location based Weather predictions with IOT sensor networks”	2022	IOT Sensor Data integration, Statistical Analysis	Fine grained sensor fusion	Limited coverage of remote areas

- The study by Muthulakshmi A and Dr. S. Baghavathi Priya focuses on the "Automatic Monitoring and Controlling of Weather Condition using Big Data Analytics." Published in 2018, it utilizes Big Data Analytics to create efficient and accurate weather forecasting models. These models are designed to predict and monitor weather datasets, specifically focusing on predicting rainfall. However, a limitation of the study is that it only predicts rainfall for past data and does not extend to forecasting future rainfall patterns.
- Smith et al. (2019) in their paper "Advancements in Location-Based Weather Forecasting" utilize Machine Learning (ML) algorithms for weather forecasting. Their approach focuses on real-time data integration, aiming for high accuracy in weather predictions. However, the method has a limitation in that it depends heavily on large datasets and involves a high model complexity, which could pose challenges in implementation and scalability.
- Brown & Jones (2020) in their work on "Image mapping for weather forecasting" utilize Convolutional Neural Networks (CNN) for image analysis in weather forecasting. Their approach aims to achieve high-resolution weather mapping and automated feature extraction from images. However, a limitation of this method is its limited interpretability, meaning that the models may not always provide clear explanations for their predictions. Additionally, the approach may be highly specialized for weather forecasting, which could limit its applicability to other domains.
- Wang et al. (2020) in their paper "Deep Learning Techniques for Location-based Weather Prediction" utilize Deep Learning methods for weather prediction based on location. Their approach is praised for its adaptability to complex weather patterns, allowing for more accurate predictions in various locations. However, a limitation of this approach is the

requirement of large datasets for training, which may pose challenges in data collection and processing, especially for locations with limited data availability.

- Johnson & Lee (2021) in their study on "Relationship Modelling for Location-based Weather Forecasting" employ Deep Learning (DL) models to model spatial-temporal relationships for weather forecasting. Their approach focuses on capturing the complex interactions between weather variables over space and time, leading to more accurate predictions. However, a limitation of their method is that it can be computationally intensive, requiring significant resources for training and inference. Additionally, DL models are often considered black-box models, meaning that the inner workings of the model are not easily interpretable, which can be a limitation in some applications where interpretability is important.
- In their work on "Probabilistic Graphical Models for Location-based Weather Forecasting," Liu & Gupta (2021) employ probabilistic graphical models for weather forecasting based on location. Their approach focuses on providing probabilistic forecasts, which are beneficial for estimating uncertainty in weather predictions. However, the method has limitations in terms of scalability, as it may struggle to handle large datasets or complex models efficiently. Additionally, parameter tuning in probabilistic graphical models can be challenging, requiring careful optimization to achieve accurate forecasts.
- Patel and Sharma (2022) in their study on "Deep Learning Approaches for Location-Based Weather Forecasting" focus on using Deep Learning techniques to extract features from satellite imagery, aiming to enhance the accuracy of weather forecasts based on location. Their approach leverages the rich information available in satellite images to improve the understanding of weather patterns. However, a limitation of their method is its dependence on the availability and quality of data, as the effectiveness of the approach relies heavily on the availability of high-quality satellite imagery for accurate feature extraction.
- David Clark and Rachel White (2022) in their work on "Enhancing Location-based Weather Predictions with IoT Sensor Networks" focus on integrating IoT sensor data into weather prediction models. Their approach involves statistical analysis of this sensor data to improve the accuracy of location-based weather predictions. One of the strengths of their approach is the fine-grained sensor fusion, which allows for a more detailed and nuanced understanding of local weather conditions. However, a limitation of this method is its limited coverage of remote areas, as IoT sensor networks may not be as extensively deployed in such regions, potentially leading to gaps in the data used for weather predictions in these areas.

CHAPTER 3
PROPOSED SYSTEM ANALYSIS

Requirement Specifications:

Both the Software and Hardware are required for this system.

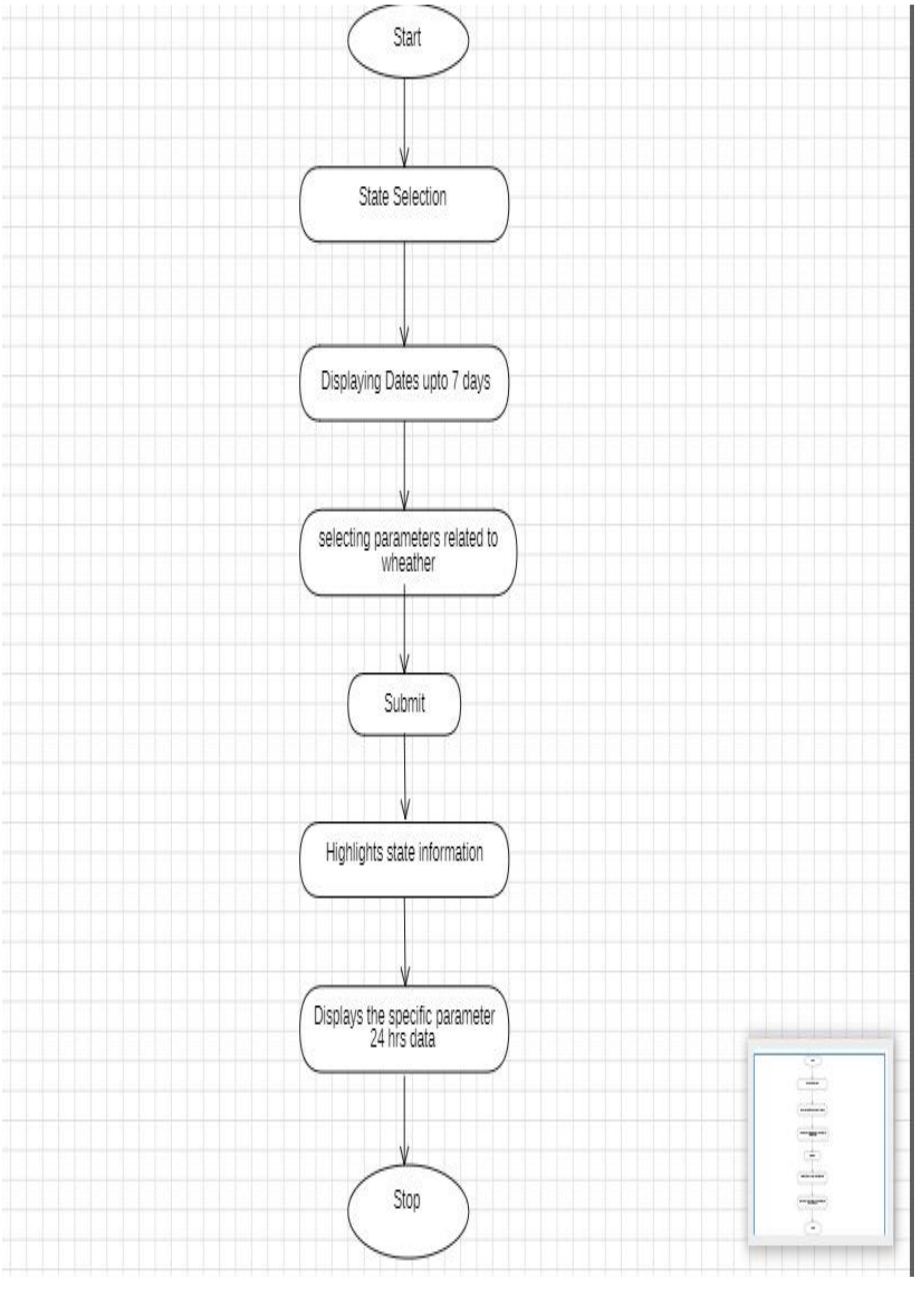
3.1 Hardware Requirements

- Processor – intel i5
- Hard Disk – 20 GB or More
- Memory – 1 GB or More

3.2 Software Requirements

- System
  - Windows 11
- IDE
  - Visual Studio Code
- Frontend
  - HTML
  - CSS
  - Java Script
- Backend
  - Python(Flask)

Flow chart


Start
State Selection
Displaying Dates upto 7 days
selecting parameters related to wheather
Submit
Highlights state information
Displays the specific parameter 24 hrs data
Stop

Flow Chart Description:

A flow chart is a graphical representation of a process, showing the steps involved and the sequence in which they occur. It typically consists of various shapes connected by arrows, indicating the flow of the process. Each shape represents a different step or action in the process.

Here the flow of working of this application is to select the initial data which is used to generate the dates for the next seven days. From the generated 7 dates select the date for which the data is to be viewed. Particularly select the name of the state and also select the weather parameter. After selection of all the fields click on the submit button which generates the red alert to highlight the selected state. Now click on the red indicator that opens a new tab with all the particular parameter data for the 24 hours. The data can be viewed in the form of images using time slider.

3.3 UML Diagrams:

- UML stands for Unified Modelling Language. UML is a standardized general purpose modelling 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.
- The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.
- The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.
- The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software project.

Contents of UML

In general, a UML diagram consists of the following features

1.**Entities**: These may be classes, objects, users or systems behaviours.

2. **Generalization**: a solid line with an arrow that points to a higher abstraction of the present Item.

3. **Association**: a solid line that represents that one entity uses another entity as part of its Behaviour.

4. **Dependency**: a dotted line with an arrowhead that shows one entity depends on the behaviour of another entity.

3.3.1. Class Diagram:

In software engineering, a class diagram in the Unified Modelling 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.

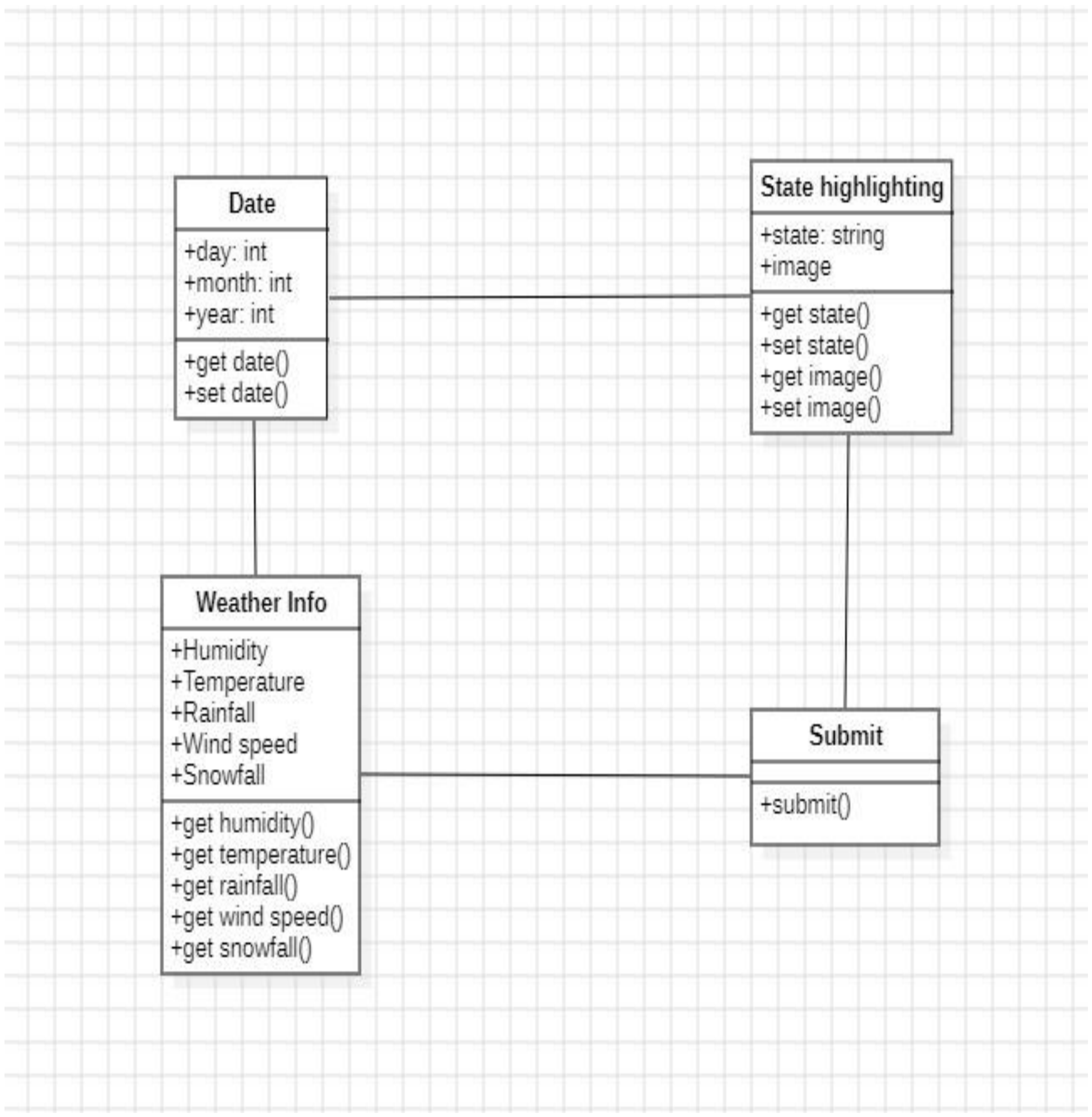


Fig: Class Diagram for State Cast-Dynamic Weather Visualization

3.3.2. Use case Diagram:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural 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.

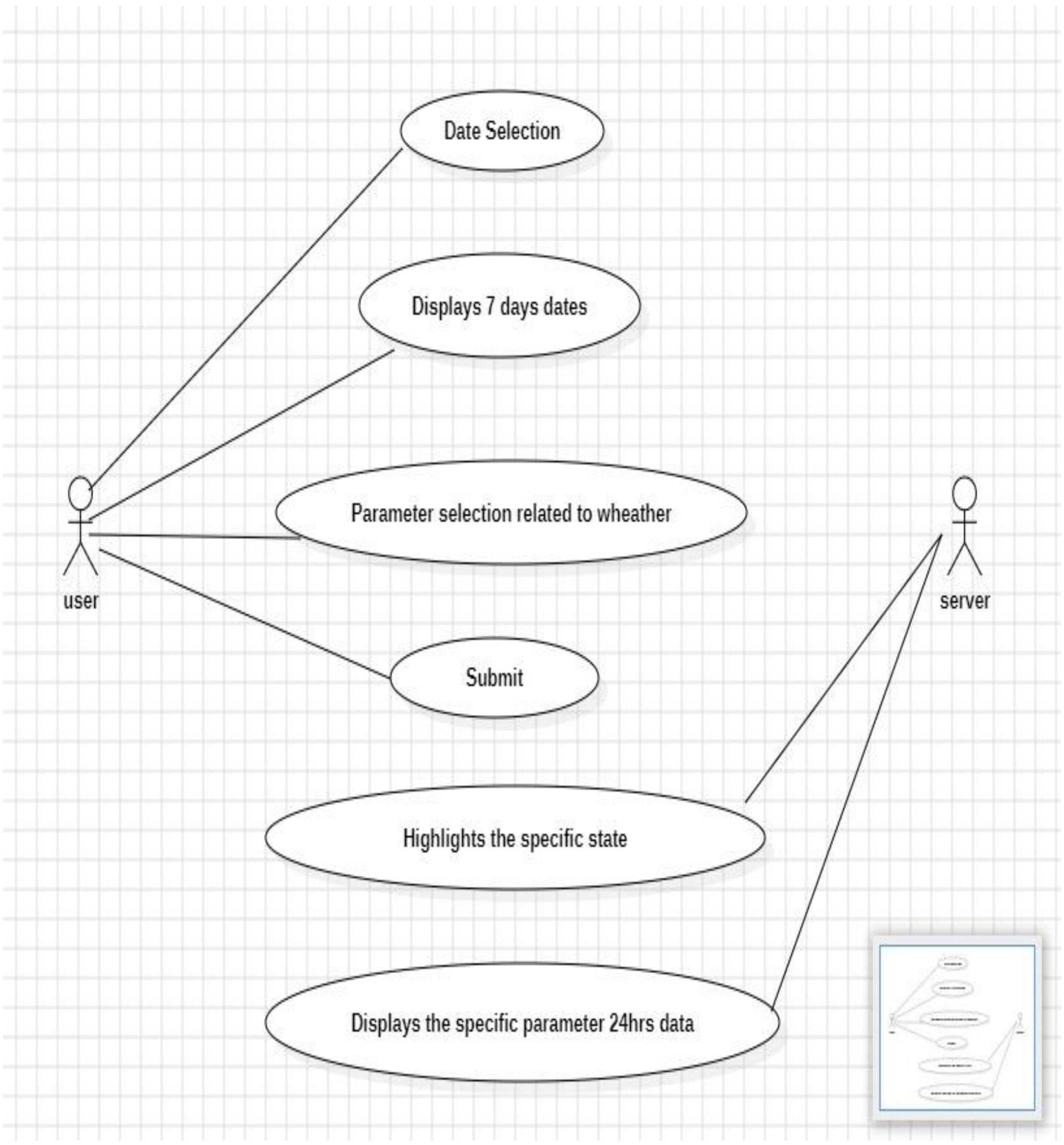

Date Selection
Displays 7 days dates
Parameter selection related to wheather
Submit
Highlights the specific state
Displays the specific parameter 24hrs data
user
server

Fig: Use case Diagram for State Cast-Dynamic Weather Visualization

3.3.3. Sequence Diagram:

A sequence diagram in Unified Modelling 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.

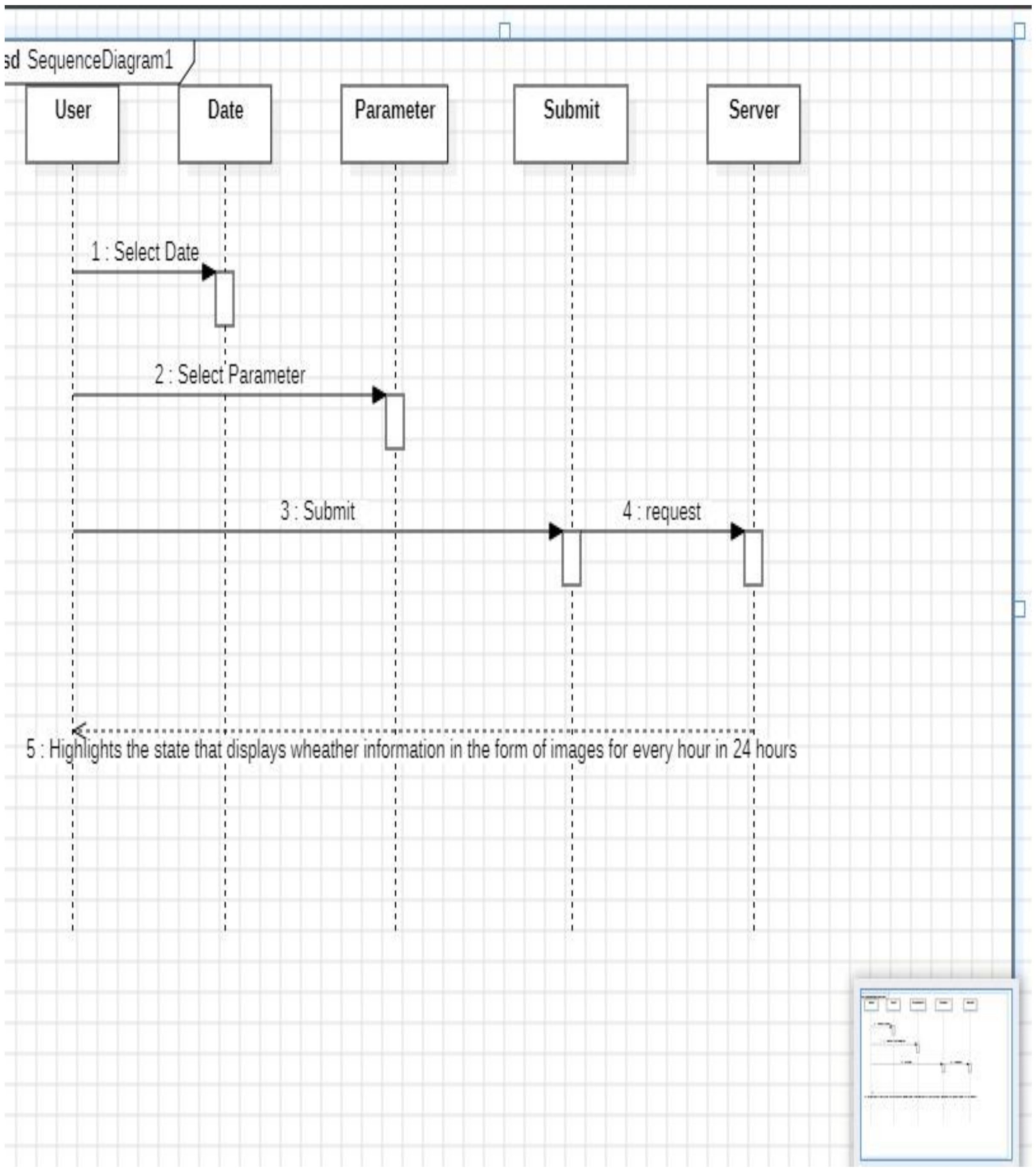

sd SequenceDiagram1
User
Date
Parameter
Submit
Server
1 : Select Date
2 : Select Parameter
3 : Submit
4 : request
5 : Highlights the state that displays wheather information in the form of images for every hour in 24 hours

Fig: Sequence Diagram for State Cast-Dynamic Weather Visualization

CHAPTER 4
SYSTEM DESIGN

Hardware Requirements

Processor:

A processor is an integrated electronic circuit that performs the calculations that run a computer. A processor performs arithmetical, logical, input/output (I/O) and other basic instructions that are passed from an operating system (OS). Most other processes are dependent on the operations of a processor.

Hard Disk:

A hard disk drive (HDD) is a fundamental component of modern computing, serving as the primary storage medium for digital data. It comprises one or more rigid disks coated with a magnetic material, storing data in binary form using magnetic charges. These disks spin at high speeds, typically ranging from 5,400 to 15,000 revolutions per minute (RPM), while a read/write head accesses and writes data onto the disk surfaces

The evolution of hard disks has been remarkable, with significant advancements in capacity, speed, and reliability. Early hard disks had limited storage capacities, often measured in megabytes (MB). However, modern hard disks can store terabytes (TB) of data, making them ideal for storing large files such as videos, photos, and software applications.

Memory:

Memory is the electronic holding place for the instructions and data a computer needs to reach quickly. It's where information is stored for immediate use. Memory is one of the basic functions of a computer, because without it, a computer would not be able to function properly.

Software Requirements

System(Windows 11):

Windows 11 is the latest version of the Windows operating system, designed to offer a more productive, secure, and connected experience. Here are the system requirements for Windows 11:

1. Processor: 1 gigahertz (GHz) or faster with at least two cores on a compatible 64-bit processor or system on a chip (SoC).

2. RAM: 4 gigabytes (GB) or more.

3. Storage: 64 GB or larger storage device.

4. System firmware: UEFI, Secure Boot capable.

5. TPM: Trusted Platform Module (TPM) version 2.0.

6. Graphics card: DirectX 12 compatible graphics / WDDM 2.x.

7. Display: >9" with HD Resolution (720p).

8. Internet connection: Internet connectivity is necessary to perform updates and to download and take advantage of some features.

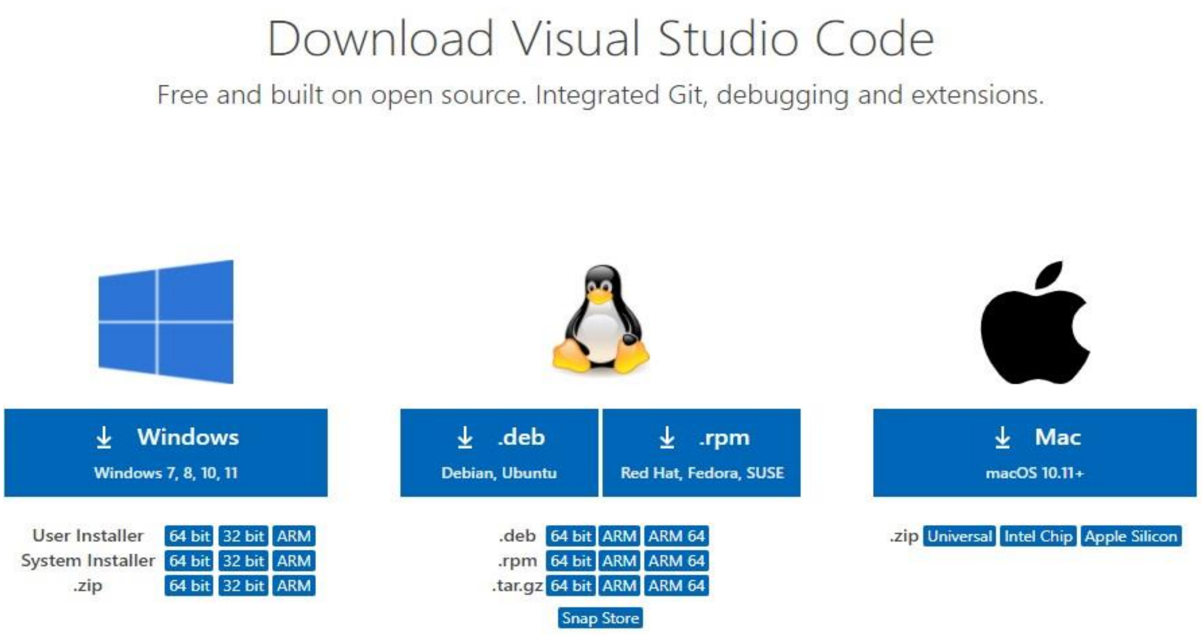
It's important to note that these are the minimum system requirements. For the best experience, especially when using advanced features or running multiple applications simultaneously, it's recommended to have higher specifications.

IDE - Visual Studio Code:

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It supports a wide range of programming languages and features built-in support for debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring. VS Code also offers extensions that can be used to customize the editor to suit different programming needs. It is widely used by developers for its flexibility, ease of use, and robust feature set.

Steps to Install Visual Studio Code on Windows Step 1:

Step 1: Visit the Official Website of the Visual Studio Code using any web browser like Google Chrome, Microsoft Edge, etc. Press the "**Download for Windows**" button on the website to start the download of the Visual Studio Code Application.


Download Visual Studio Code

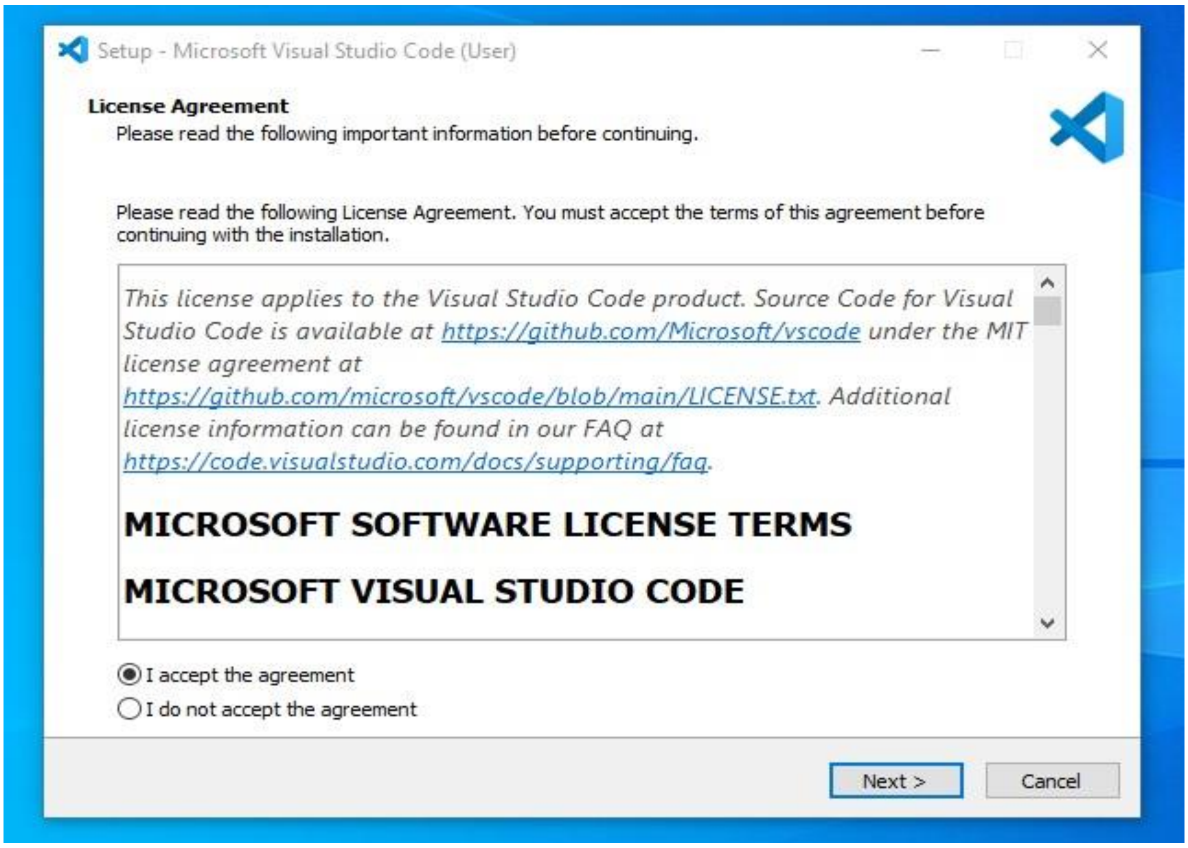
Free and built on open source. Integrated Git, debugging and extensions.

Fig – 4.1: Download Visual Studio Code

Step - 2: When the download finishes, then the **Visual Studio Code Icon** appears in the downloads folder.

Step - 3: Click on the **Installer** icon to start the installation process of the Visual Studio Code. After the Installer opens, it will ask you to accept the terms and conditions of the Visual Studio Code.

Step – 4: Click on **I accept the agreement** and then click the **Next** button.



Setup - Microsoft Visual Studio Code (User)

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at https://github.com/Microsoft/vscode under the MIT license agreement at https://github.com/microsoft/vscode/blob/main/LICENSE.txt. Additional license information can be found in our FAQ at https://code.visualstudio.com/docs/supporting/faq.

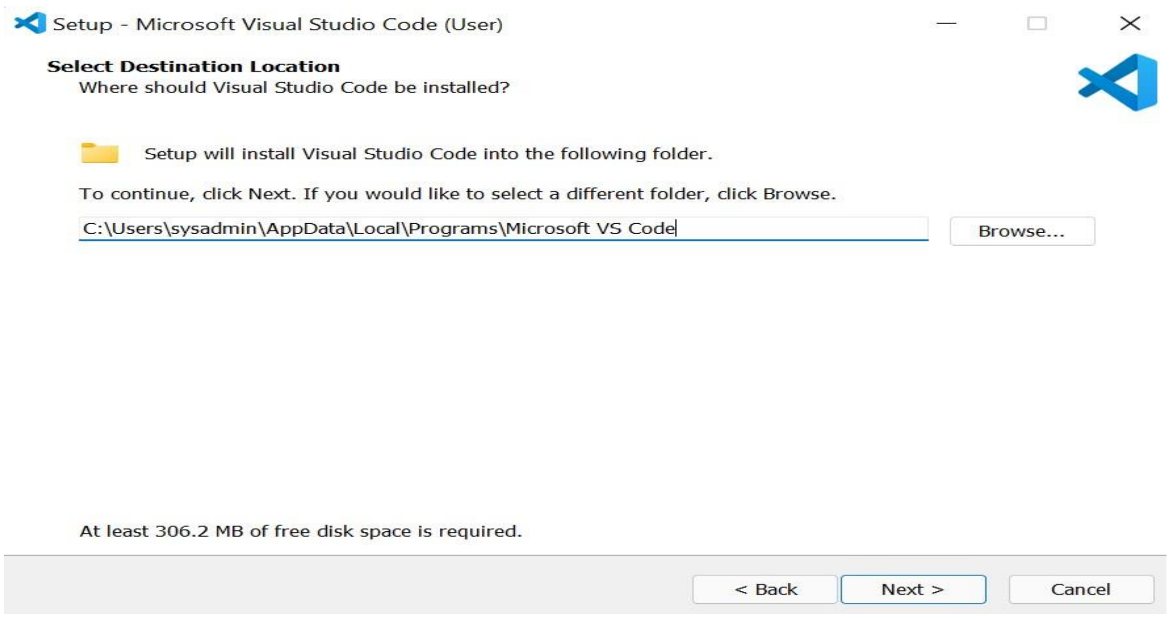
MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL STUDIO CODE

● I accept the agreement
○ I do not accept the agreement

Fig – 4.2: License Agreement

Step – 5: Select the drive location where you need to install the VS Code.



Setup - Microsoft Visual Studio Code (User)

Select Destination Location

Where should Visual Studio Code be installed?

Setup will install Visual Studio Code into the following folder.

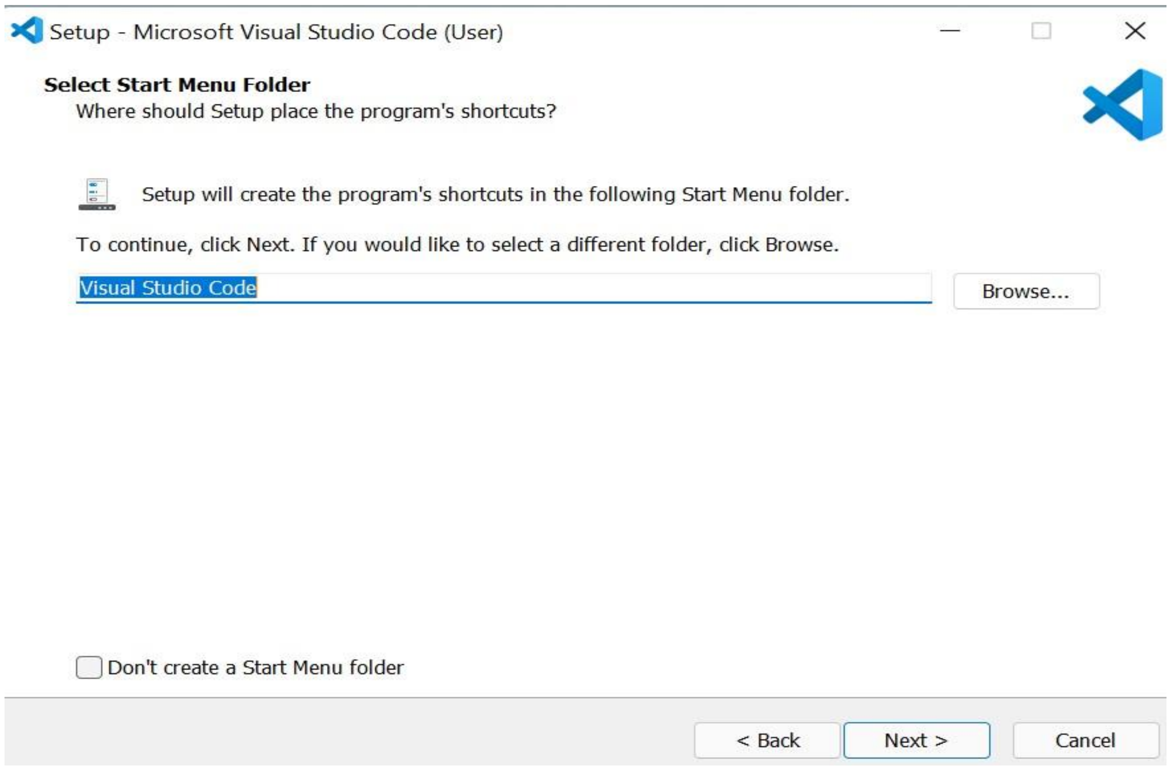
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Users\sysadmin\AppData\Local\Programs\Microsoft VS Code

At least 306.2 MB of free disk space is required.

Fig – 4.3: Select Path

Step 6: Select the folder name for the VS Code setup files.



Setup - Microsoft Visual Studio Code (User)

Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

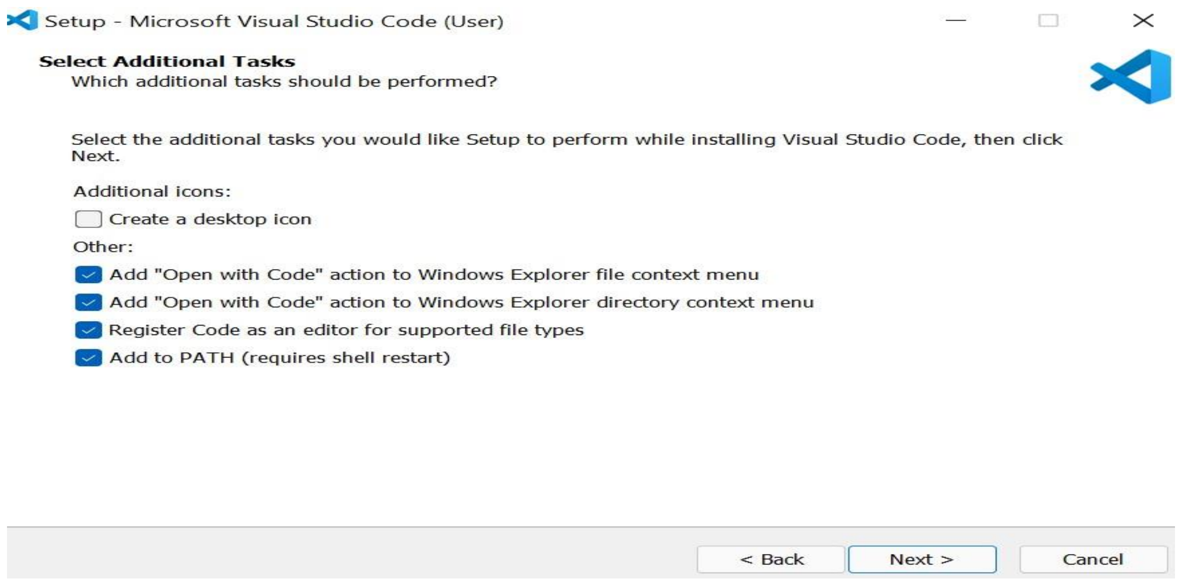
To continue, click Next. If you would like to select a different folder, click Browse.

Visual Studio Code

☐ Don't create a Start Menu folder

Fig – 4.4: Select Folder Name

Step – 7: Select the additional task, you want to do by the installer. I wanted to have "Open with Code" added to both the File and Directory content menu, hence I selected those.



Select Additional Tasks
Which additional tasks should be performed?

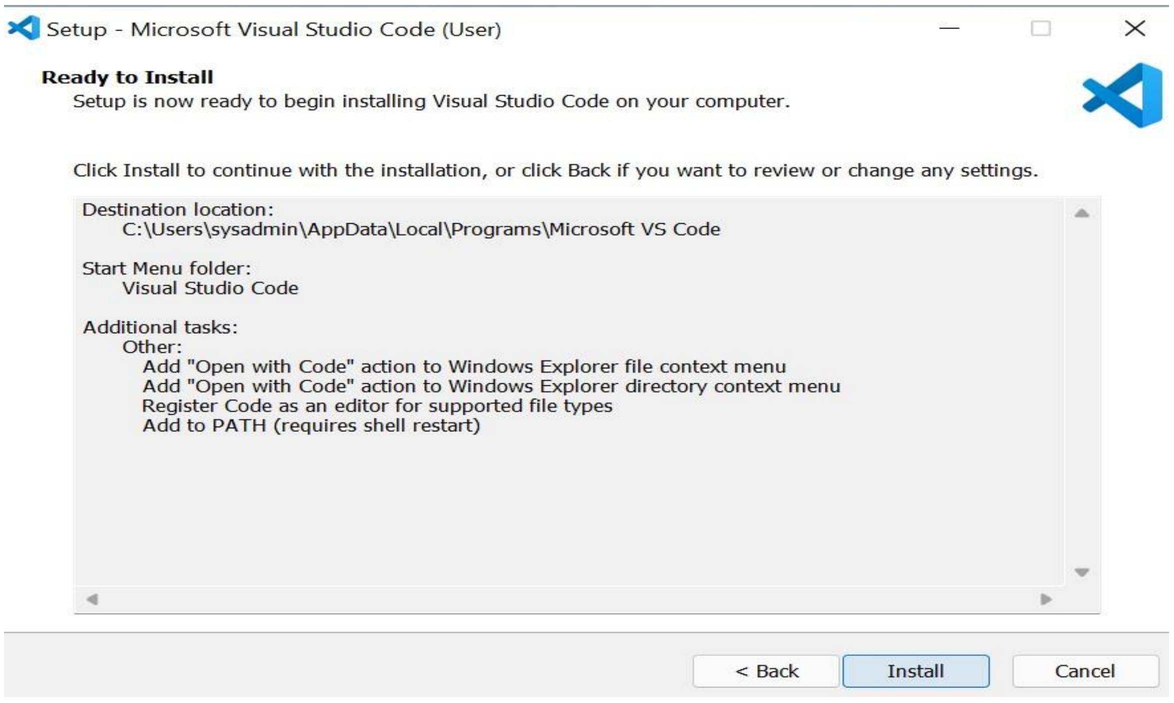
Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:
- [ ] Create a desktop icon

Other:
- [✓] Add "Open with Code" action to Windows Explorer file context menu
- [✓] Add "Open with Code" action to Windows Explorer directory context menu
- [✓] Register Code as an editor for supported file types
- [✓] Add to PATH (requires shell restart)

Fig – 4.5: Select Additional Task

Step 8: Check if all the settings. Click on "< Back" if you need to change anything.



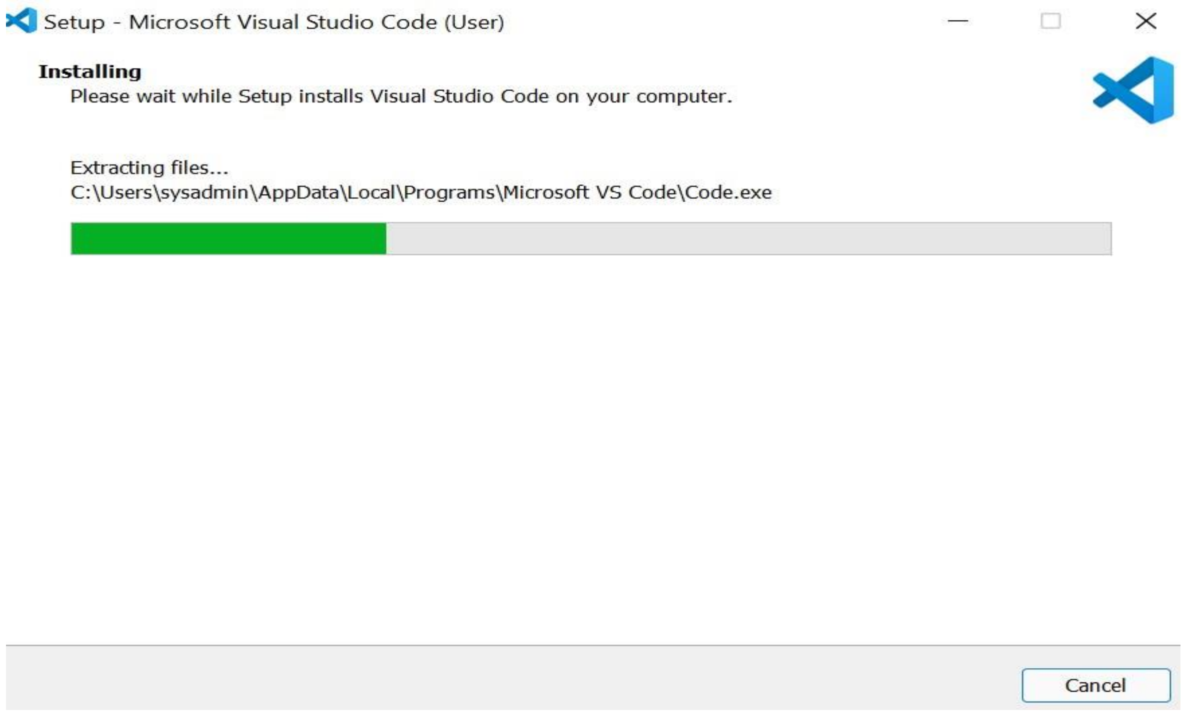
Ready to Install
Setup is now ready to begin installing Visual Studio Code on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
      C:\Users\sysadmin\AppData\Local\Programs\Microsoft VS Code

Start Menu folder:
      Visual Studio Code

Additional tasks:
      Other:
         Add "Open with Code" action to Windows Explorer file context menu
         Add "Open with Code" action to Windows Explorer directory context menu
         Register Code as an editor for supported file types
         Add to PATH (requires shell restart)
```

Fig – 4.6: Ready to Install

Step – 9: Click on "Install", and the installation process will start.

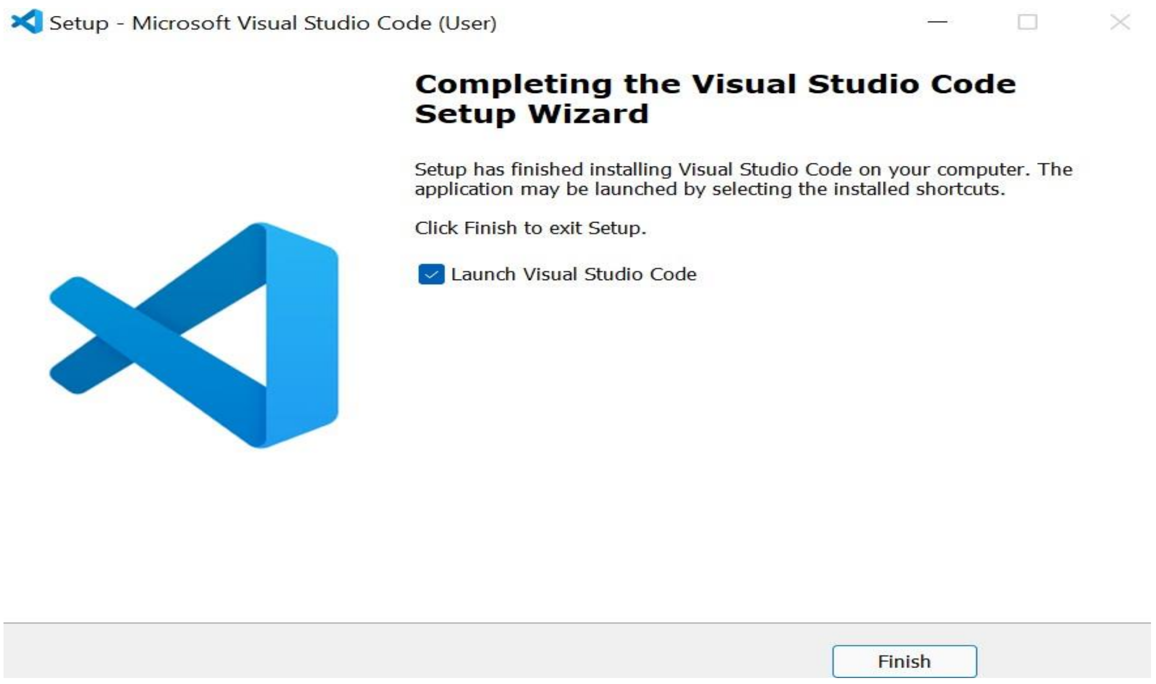

Setup - Microsoft Visual Studio Code (User)

Installing
Please wait while Setup installs Visual Studio Code on your computer.

Extracting files...
C:\Users\sysadmin\AppData\Local\Programs\Microsoft VS Code\Code.exe

Fig – 4.7: Installation

Step 10: After installation, click "Finish"


Setup - Microsoft Visual Studio Code (User)

Completing the Visual Studio Code Setup Wizard

Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

[✓ Launch Visual Studio Code]

Fig – 4.8: Launch Visual Studio Code

Step – 11: After the previous step, the **Visual Studio Code window** opens successfully. Now you can create a new file in the Visual Studio Code.

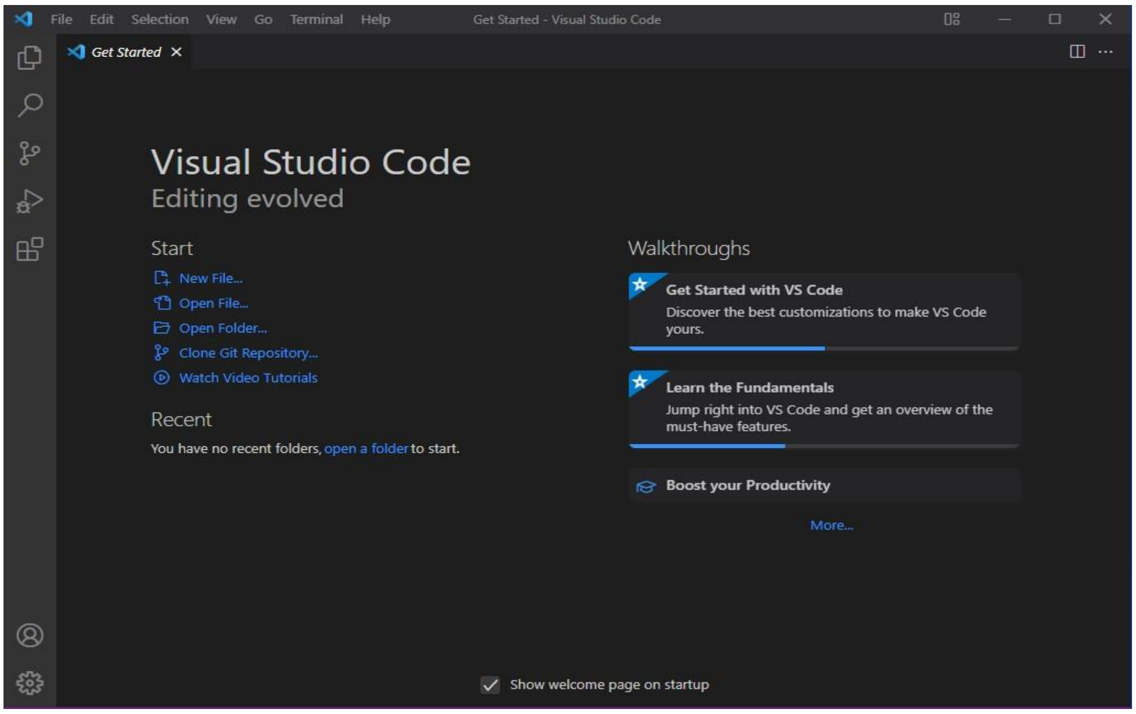


Fig – 4.9: Visual Studio Code window

Frontend:

a) <u>HTML(Hyper Text Markup Language):</u>

HTML is the standard markup language used to create and design web pages. It defines the structure of content on a web page using a system of tags and attributes. HTML tags are used to mark up elements such as headings, paragraphs, lists, links, images, and more, while attributes provide additional information about these elements. HTML is the foundation of web development and is used alongside CSS (Cascading Style Sheets) and JavaScript to create interactive and visually appealing websites.

Features of HTML:

1. **Simple and Easy to Learn**:
   HTML has a straightforward syntax and is easy for beginners to grasp.
2. **Platform Independent**:
   HTML can be viewed on any platform, including Windows, macOS, Linux, and mobile devices
3. **Supports Multimedia**:
   HTML supports embedding images, audio, video, and other multimedia elements into web pages.

4. **Hyperlinking:**
   HTML allows the creation of hyperlinks, enabling users to navigate between different web pages.
5. **Semantic Elements**:
   HTML5 introduces semantic elements like <header>, <footer>, <article>, <section>, <nav>, and <aside>, which provide a better structure meaning to web content for search engines and screen readers.
6. **Forms:**
   HTML provides form elements like <input>, <textarea>, <select>, <button> etc., for user input, making it possible to collect data from users.
7. **Accessibility:**
   HTML supports accessibility features, such as alt text for images and semantic elements, making web content more accessible to people with disabilities.
8. **Integration with CSS and JavaScript**:
   HTML works seamlessly with CSS for styling and JavaScript for interactivity, enabling the creation of dynamic and visually appealing web pages.

Applications of HTML:

1. **Website Development:**
   HTML is used to create the structure and content of websites.
2. **Email Templates:**
   HTML is used to design email templates for marketing campaigns, newsletters, and personal emails.
3. **Web Applications**:
   HTML is used in conjunction with CSS and JavaScript to develop interactive web applications.
4. **E-learning:**
   HTML is used to create e-learning courses and platforms.
5. **Document Structure:**
   HTML is used to create the structure and layout of online documents, such as articles, reports, and manuals.
6. **Responsive Web Design:**
   HTML is used to create responsive web designs that adapt to different screen sizes and devices.
7. **Search Engine Optimization (SEO):**
   Proper use of HTML elements and attributes can improve the search engine ranking of web pages.

b) <u>CSS(Cascading Style Sheets):</u>

CSS is a style sheet language used for describing the presentation of a document written in a markup language like HTML. It defines how elements should be displayed on screen, in print, or spoken aloud. CSS can control the layout, colors, fonts, and other visual aspects of a webpage.

Features of CSS:

1. **Selectors:**
   CSS selectors are patterns used to select the elements you want to style. They can select elements based on their tag name, class, ID, attributes, and more.
2. **Box Model:**
   CSS uses a box model to represent elements on a webpage. It includes properties like width, height, padding, border, and margin, which allow you to control the size and spacing of elements.
3. **Layout:**
   CSS provides different layout techniques, including flexbox and grid, which allow you to create complex and responsive layouts.
4. **Typography**:
   CSS allows you to control the font size, style, weight, and alignment of text on a webpage.
5. **Colors and Backgrounds:**
   CSS provides properties to set the color, background color, and background image of elements.
6. **Animations and Transitions:**
   CSS can be used to create animations and transitions to add interactivity and visual effects to a webpage.

Applications of CSS:

1. **Web Design:**
   CSS is primarily used for styling websites, including layout, colors, typography, and visual effects.
2. **Responsive Design**:
   CSS is essential for creating responsive websites that adapt to different screen sizes and devices.
3. **Print Styling**:
   CSS can be used to create print-friendly versions of web pages by specifying styles for printing.
4. **Accessibility**:
   CSS can be used to improve the accessibility of a website by styling elements in a way that is easy to read and navigate for all users.
5. **Cross-Browser Compatibility**:
   CSS helps in ensuring that a website looks and behaves consistently across different web browsers.
6. **Interactive Web Elements**:
   CSS can be used to create interactive elements like buttons, menus, and forms using styles and animations.

c) <u>Java Script:</u>

JavaScript is a high-level, dynamic, and interpreted programming language. It is widely used for creating interactive and dynamic websites. JavaScript code can be embedded directly into HTML pages and executed by web browsers. It is also commonly used in server-side development (Node.js), game development, and mobile app development (React Native, Ionic).

Features of JavaScript:

1. **Client-Side Scripting**:
   JavaScript code runs on the client's browser, enabling interactive web pages without server interaction.
2. **Event-Driven Programming**:
   JavaScript allows developers to create code that responds to user actions like clicks, scrolls, and inputs.
3. **Asynchronous Programming**:
   JavaScript supports asynchronous operations, allowing tasks to run independently without blocking the main thread.
4. **Dynamic Typing**:
   Variables in JavaScript are not explicitly typed, making it flexible but requiring careful handling to avoid errors.
5. **Object-Oriented**:
   JavaScript supports object-oriented programming paradigms, allowing the creation of objects and classes.
6. **Cross-Platform Compatibility**:
   JavaScript can run on various platforms, including web browsers, servers (Node.js), and mobile devices.
7. **Extensibility:**
   JavaScript can be extended using libraries and frameworks like React, Angular, and Vue.js to build complex applications.
8. **Functional Programming**:
   JavaScript supports functional programming concepts, allowing functions to be treated as first-class citizens.

Applications of JavaScript:

1. **Web Development**:
   JavaScript is primarily used for enhancing the user interface and adding interactivity to websites.
2. **Mobile App Development**:
   JavaScript frameworks like React Native and Ionic allow developers to cross-platform mobile applications
3. **Game Development**:
   JavaScript is used with libraries like Phaser and Three.js to create browser-based
4. **Server-Side Development:**
   Node.js enables developers to use JavaScript for server-side programming, handling requests and managing databases.

5. **Web APIs:**
   JavaScript is used to interact with web APIs, enabling features like geolocation, camera access, and notifications in web applications.

Backend

Python:

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python is versatile and can be used for various types of programming, including web development, data analysis, artificial intelligence, scientific computing, and more. Here are some key features and applications of Python:

Features of Python:

1. **Easy to Read and Write**:
   Python's syntax is designed to be readable and clean, making it easier to write and understand code.
2. **Interpreted Language:**
   Python code is executed line by line by the Python interpreter, making it easy to test code quickly.
3. **Dynamic Typing:**
   Python is dynamically typed, meaning you don't need to specify variable types. This makes Python flexible and allows for faster development.
4. **Cross-Platform:**
   Python is available on multiple platforms (Windows, macOS, Linux), making it versatile for different operating systems.
5. **Large Standard Library:**
   Python comes with a large standard library that provides support for many common programming tasks, such as file I/O, networking, and web services.
6. **Object-Oriented:**
   Python supports object-oriented programming (OOP) principles, allowing you to create reusable and modular code.
7. **Extensible:**
   Python can be extended by adding new modules implemented in other languages, such as C or C++, to optimize performance or access system resources.
8. **Community Support:**
   Python has a large and active community of developers who contribute to its development and provide support through forums and online resources.

Applications of Python:

1. **Web Development**:
   Python is used with web frameworks like Django and Flask to build dynamic and scalable web applications.
2. **Data Science and Machine Learning**:
   Python is widely used in data analysis, machine learning, and artificial intelligence due to libraries like NumPy, pandas, and scikit-learn.

3. Scientific Computing:
Python is used in scientific computing and numerical analysis with libraries like SciPy and matplotlib.

4. Automation and Scripting:
Python is used for automating repetitive tasks and writing scripts for various purposes, such as system administration and testing.

5. Game Development:
Python is used in game development with libraries like Pygame for creating 2D games.

6. Desktop GUI Applications:
Python can be used to create desktop GUI applications using libraries like Tkinter, PyQt, and wxPython.

7. Web Scraping:
Python is used for web scraping, extracting data from websites, and processing it for analysis or other purposes.

Overall, Python's versatility, readability, and extensive libraries make it a popular choice for developers across various domains.

Flask:

Flask is a lightweight web framework for Python, designed to make web development simple and easy. It is classified as a micro-framework because it does not require any specific tools or libraries, allowing developers to use the libraries they prefer.

Features of Flask:

1**. Lightweight and Modular**:
Flask is minimalistic and allows developers to add only the components they need, making it lightweight and efficient.

2**. Easy to Get Started:**
Flask is easy to set up and requires minimal configuration, making it ideal for beginners and small projects.

3. Flexible:
Flask provides a lot of flexibility, allowing developers to use any ORM, template engine, or other libraries they prefer.

4. Jinja2 Templating:
Flask uses the Jinja2 templating engine, which provides powerful features for creating dynamic web pages.

5. Built-in Development Server:
Flask comes with a built-in development server, making it easy to test and debug applications.

6. RESTful Request Dispatching:
Flask supports RESTful request dispatching, making it easy to build RESTful APIs.

7. Extensions:
Flask has a rich ecosystem of extensions that add additional functionality, such as authentication, database integration, and form validation.

8. Unicode -based:
Flask natively supports Unicode, making it easy to develop multilingual web applications.

9. Werkzeug:
Flask is built on the Werkzeug WSGI toolkit, which provides a solid foundation for handling HTTP requests and responses.

Applications of Flask:

1.Web Development:

Flask is used to build web applications, ranging from simple websites to complex web applications.

2.API Development:

Flask is commonly used to build RESTful APIs for web and mobile applications.

3. **Prototyping:**

Flask is often used for prototyping and developing proof-of-concept applications quickly.

4. **Microservices:**

Flask is well-suited for building microservices due to its lightweight and modular nature.

5**. Educational Purposes:**

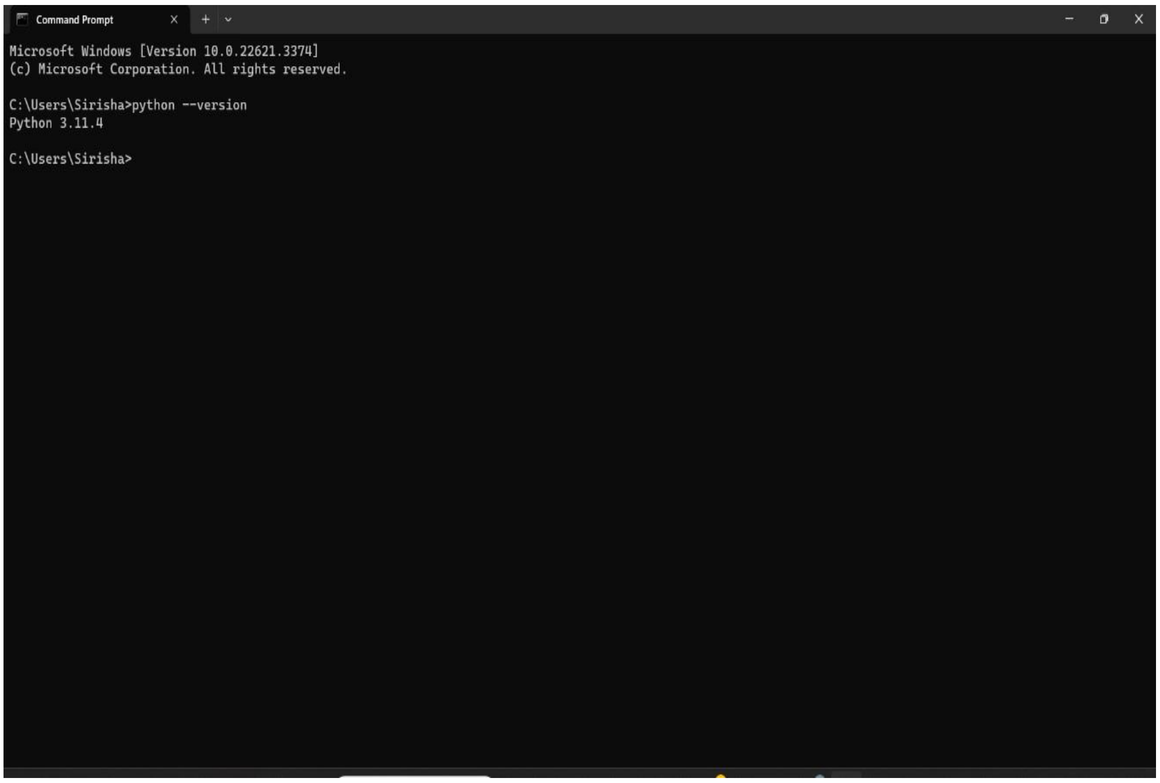
Flask is popular for teaching web development and Python programming due to its simplicity and ease of use.

Installation of Flask

Step – 1: Prerequisites

Before installing Flask, ensure that you have Python installed on your system.

You can download and install Python from the official Python website: python.org.

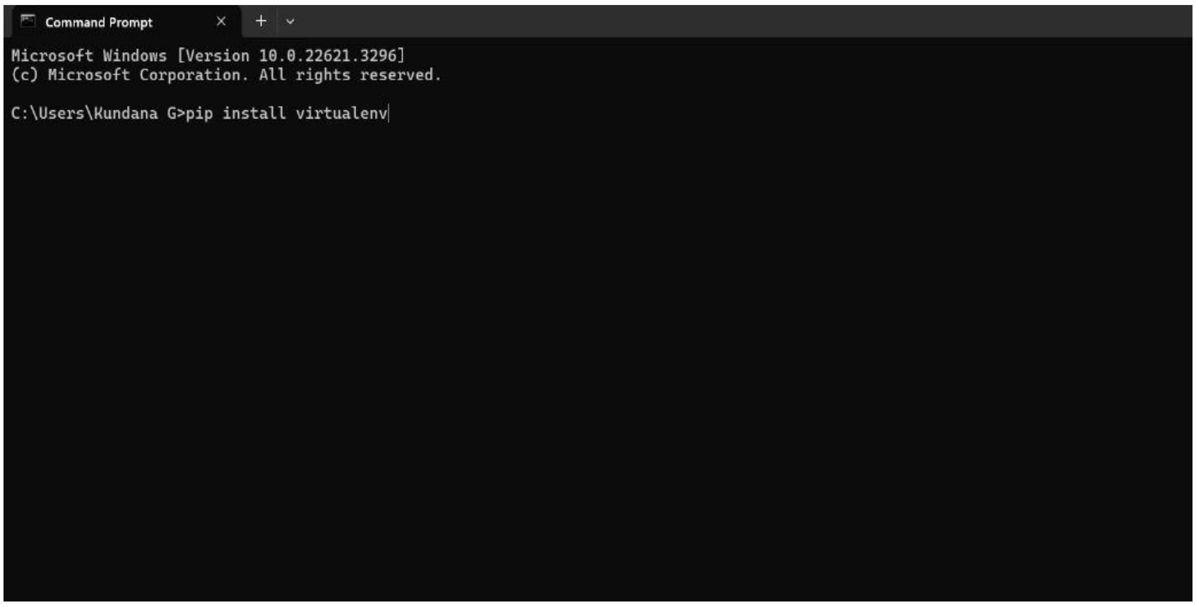


```
Microsoft Windows [Version 10.0.22621.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sirisha>python --version
Python 3.11.4

C:\Users\Sirisha>
```

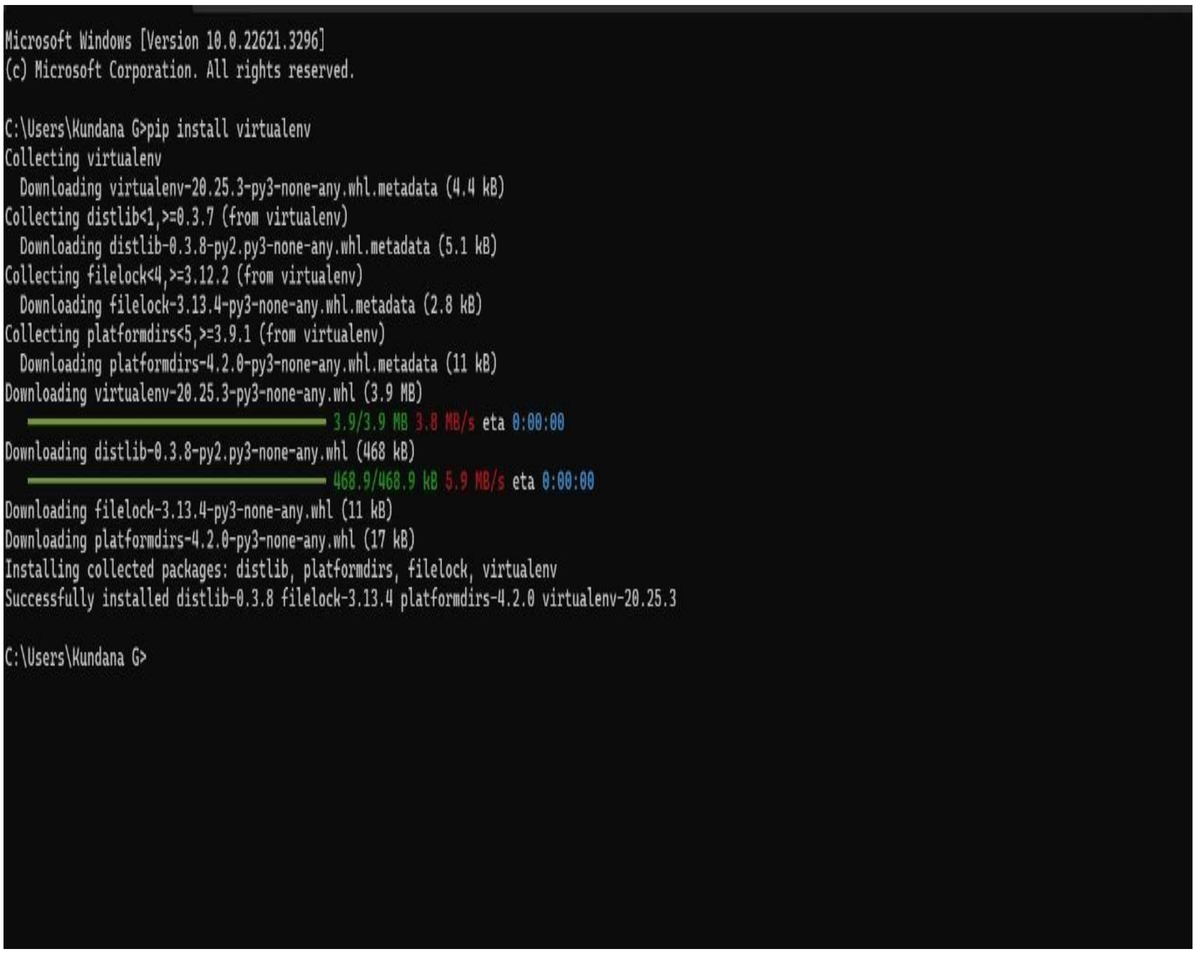
Fig:4.1.1-Checking python version

Step – 2: Virtual Environment

After installing python, create a virtual environment for your Flask projects to manage dependencies and isolate them from system-wide installations. We use a module named virtualenv which is a tool to create isolated Python environments. Virtualenv creates a folder that contains all the necessary executables to use the packages that a Python project would need. To create a virtual environment, open your command line or terminal and run the following command:



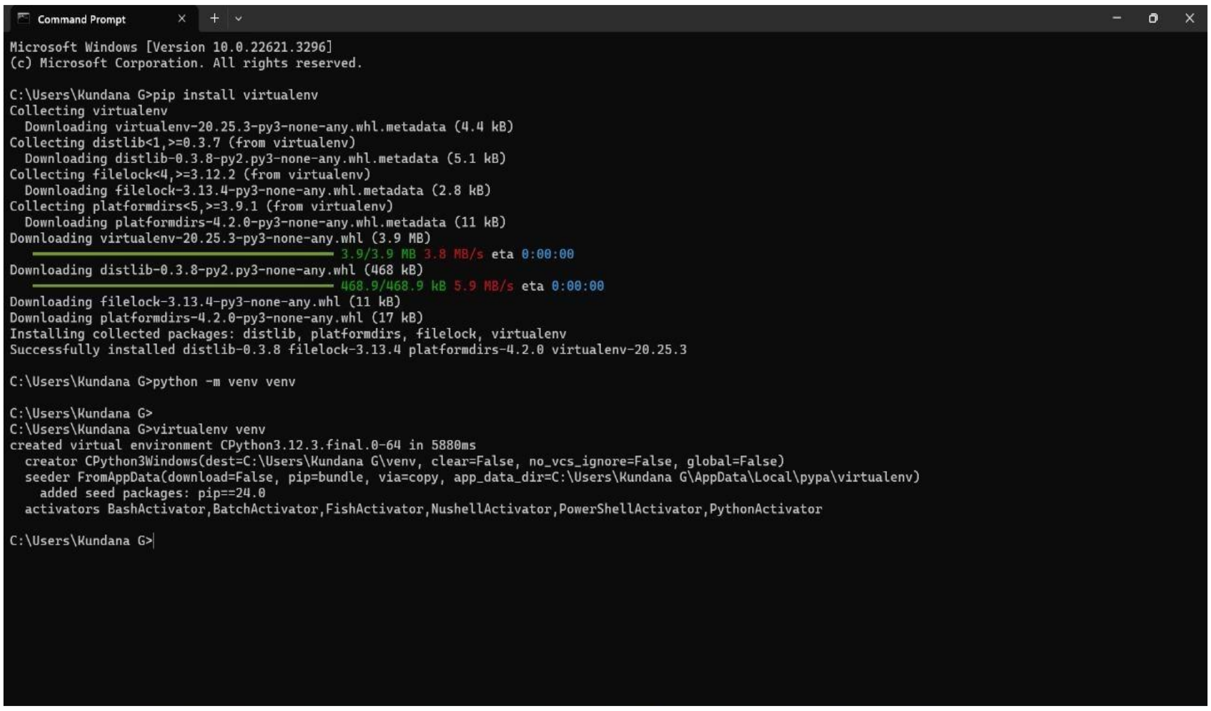
```
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kundana G>pip install virtualenv
```



```
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kundana G>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.25.3-py3-none-any.whl.metadata (4.4 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Downloading filelock-3.13.4-py3-none-any.whl.metadata (2.8 kB)
Collecting platformdirs<5,>=3.9.1 (from virtualenv)
  Downloading platformdirs-4.2.0-py3-none-any.whl.metadata (11 kB)
Downloading virtualenv-20.25.3-py3-none-any.whl (3.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.9/3.9 MB 3.8 MB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 468.9/468.9 kB 5.9 MB/s eta 0:00:00
Downloading filelock-3.13.4-py3-none-any.whl (11 kB)
Downloading platformdirs-4.2.0-py3-none-any.whl (17 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.8 filelock-3.13.4 platformdirs-4.2.0 virtualenv-20.25.3

C:\Users\Kundana G>
```

Fig-4.1.2:Install Virtual Environment

python -m venv myenv



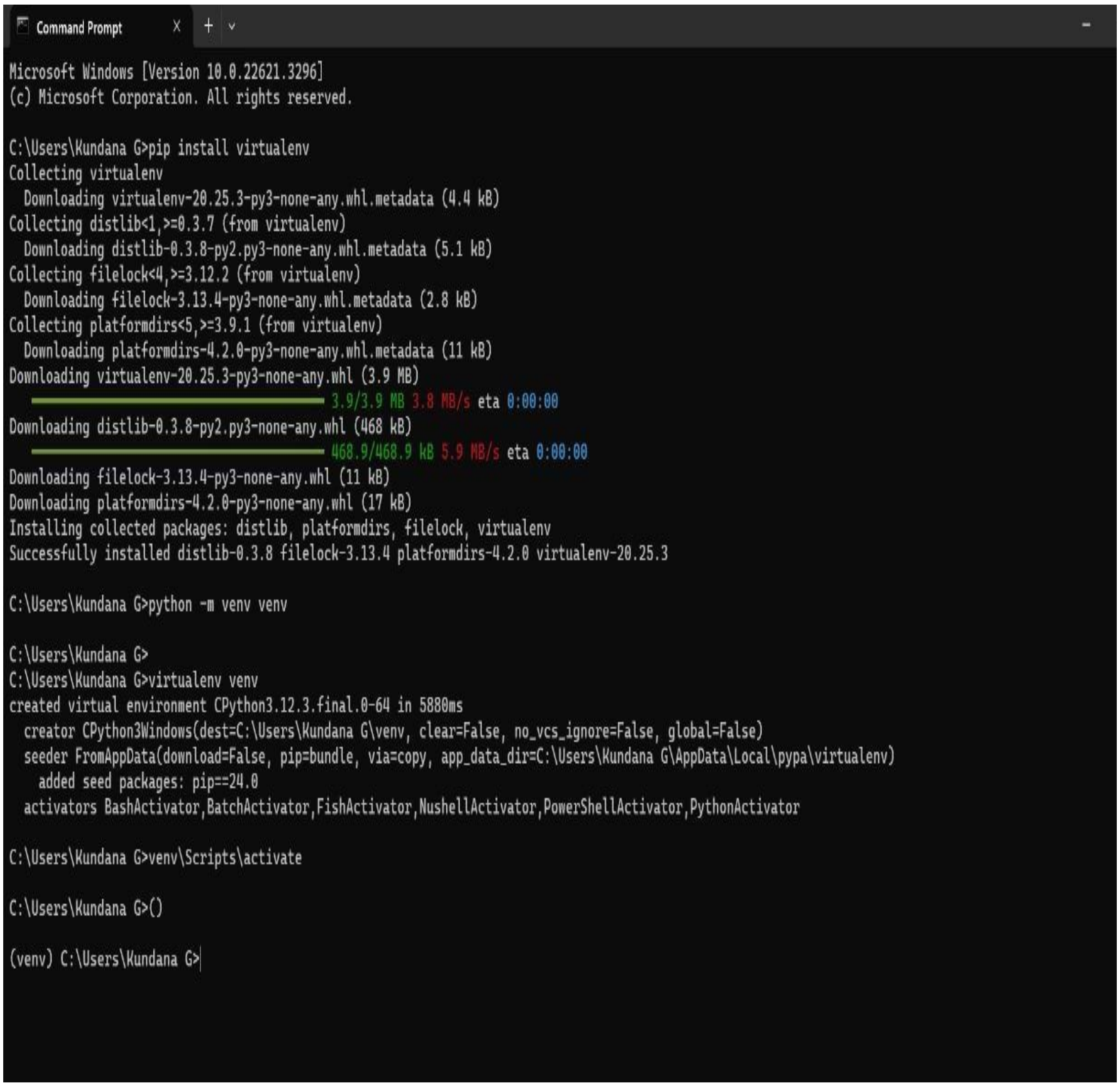
```
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kundana G>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.25.3-py3-none-any.whl.metadata (4.4 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Downloading filelock-3.13.4-py3-none-any.whl.metadata (2.8 kB)
Collecting platformdirs<5,>=3.9.1 (from virtualenv)
  Downloading platformdirs-4.2.0-py3-none-any.whl.metadata (11 kB)
Downloading virtualenv-20.25.3-py3-none-any.whl (3.9 MB)
   ---------------------------------------- 3.9/3.9 MB 3.8 MB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
   ---------------------------------------- 468.9/468.9 kB 5.9 MB/s eta 0:00:00
Downloading filelock-3.13.4-py3-none-any.whl (11 kB)
Downloading platformdirs-4.2.0-py3-none-any.whl (17 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.8 filelock-3.13.4 platformdirs-4.2.0 virtualenv-20.25.3

C:\Users\Kundana G>python -m venv venv

C:\Users\Kundana G>
C:\Users\Kundana G>virtualenv venv
created virtual environment CPython3.12.3.final.0-64 in 5880ms
  creator CPython3Windows(dest=C:\Users\Kundana G\venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\Kundana G\AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\Kundana G>
```

Fig-4.1.3:Create Path for Virtual Environment

Step – 3: Activate the Virtual Environment (Windows):

After creating the virtual environment, we need to activate it. To activate the virtual environment, run the following command in the terminal:

```
venv\Scripts\activate
```



Command Prompt

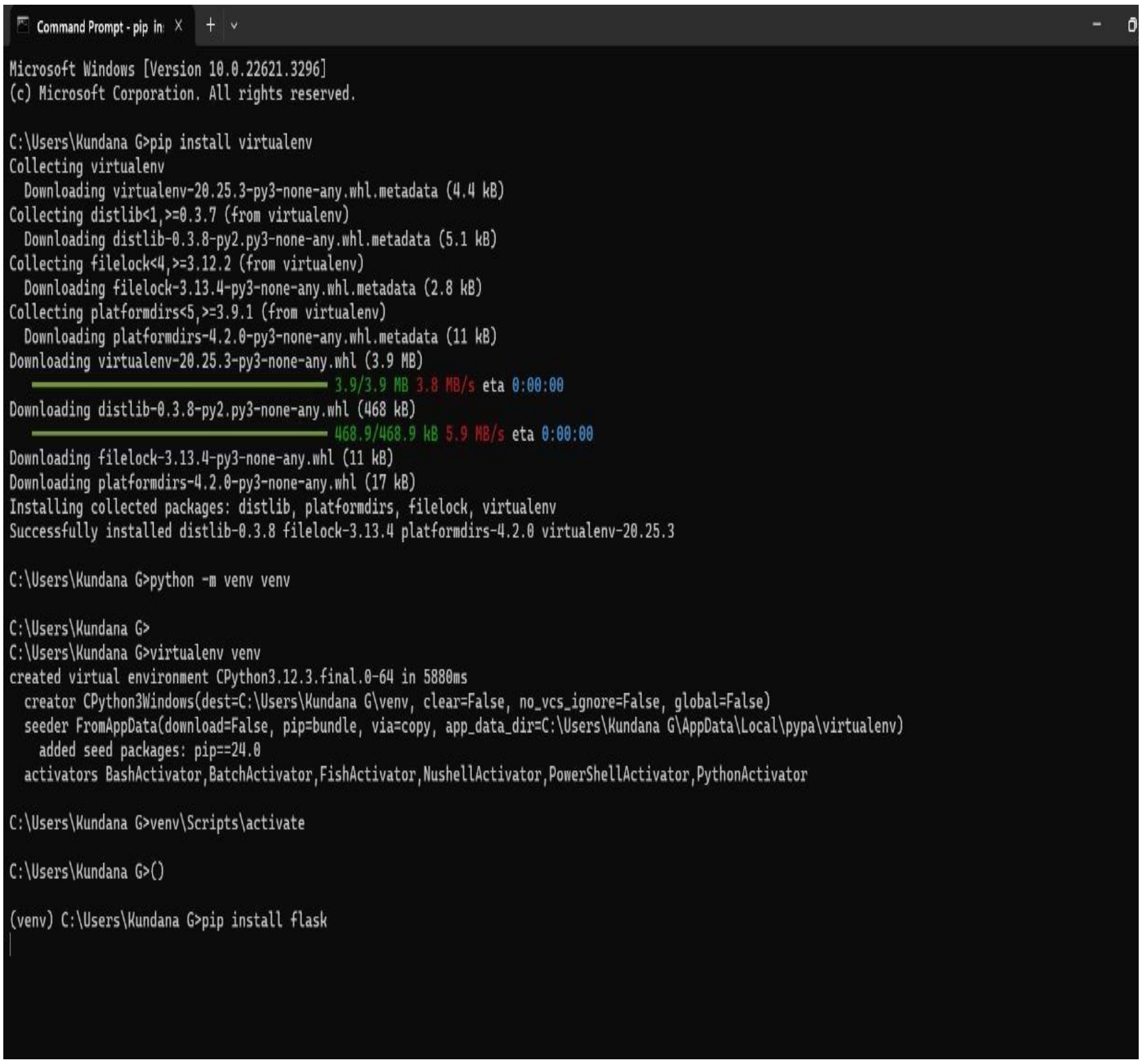
```
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kundana G>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.25.3-py3-none-any.whl.metadata (4.4 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Downloading filelock-3.13.4-py3-none-any.whl.metadata (2.8 kB)
Collecting platformdirs<5,>=3.9.1 (from virtualenv)
  Downloading platformdirs-4.2.0-py3-none-any.whl.metadata (11 kB)
Downloading virtualenv-20.25.3-py3-none-any.whl (3.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.9/3.9 MB 3.8 MB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 468.9/468.9 kB 5.9 MB/s eta 0:00:00
Downloading filelock-3.13.4-py3-none-any.whl (11 kB)
Downloading platformdirs-4.2.0-py3-none-any.whl (17 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.8 filelock-3.13.4 platformdirs-4.2.0 virtualenv-20.25.3

C:\Users\Kundana G>python -m venv venv

C:\Users\Kundana G>
C:\Users\Kundana G>virtualenv venv
created virtual environment CPython3.12.3.final.0-64 in 5880ms
  creator CPython3Windows(dest=C:\Users\Kundana G\venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\Kundana G\AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\Kundana G>venv\Scripts\activate

C:\Users\Kundana G>()

(venv) C:\Users\Kundana G>
```

Fig-4.1.4:Activate Virtual Environment

Step – 4: Install Flask: Once your virtual environment is activated, you can install Flask using pip, Python's package manager. To install Flask, run the following command in the terminal:

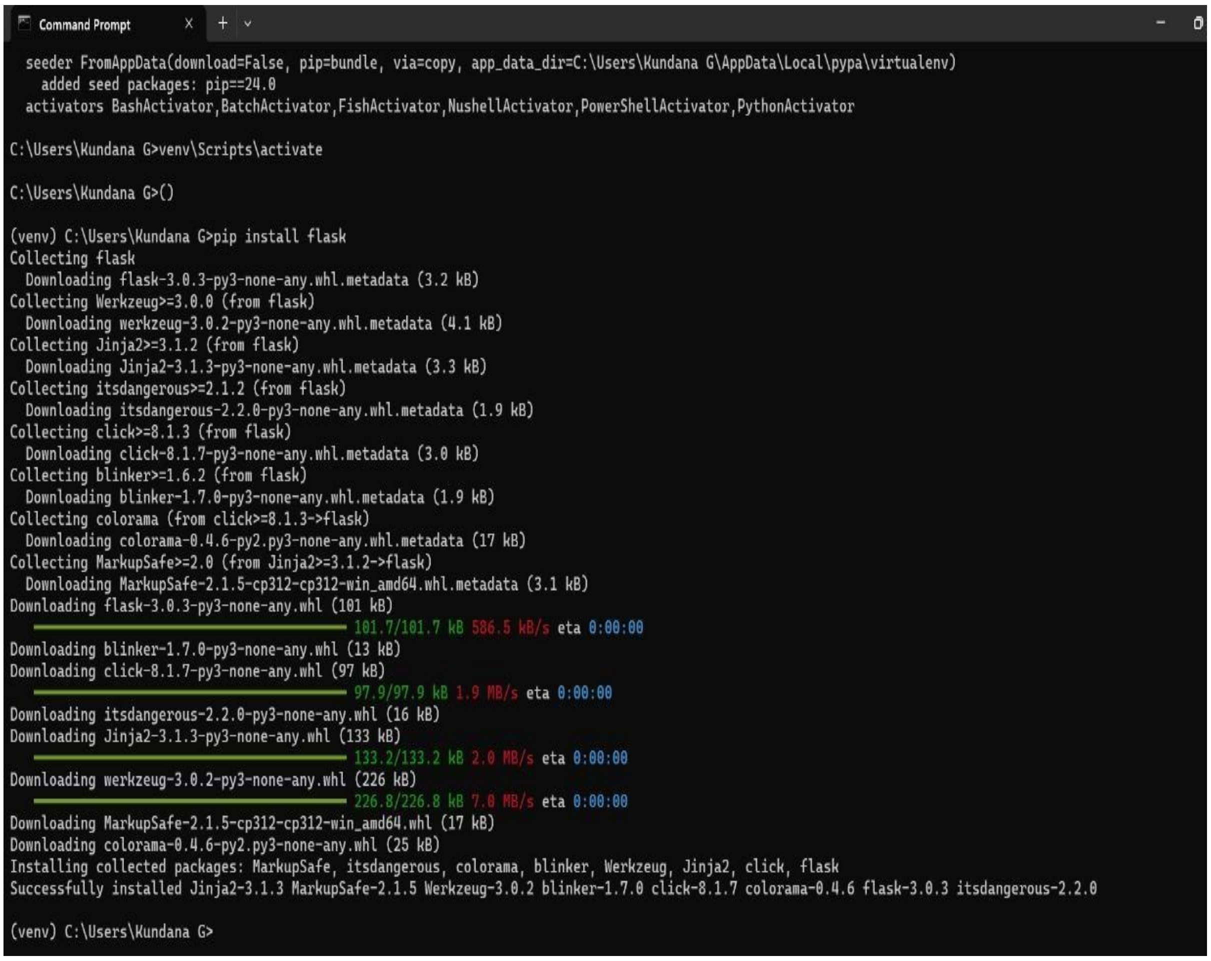
```
pip install flask
```

This command will download and install Flask and all its dependencies in your virtual environment.



```
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Kundana G>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.25.3-py3-none-any.whl.metadata (4.4 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Collecting filelock<4,>=3.12.2 (from virtualenv)
  Downloading filelock-3.13.4-py3-none-any.whl.metadata (2.8 kB)
Collecting platformdirs<5,>=3.9.1 (from virtualenv)
  Downloading platformdirs-4.2.0-py3-none-any.whl.metadata (11 kB)
Downloading virtualenv-20.25.3-py3-none-any.whl (3.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.9/3.9 MB 3.8 MB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 468.9/468.9 kB 5.9 MB/s eta 0:00:00
Downloading filelock-3.13.4-py3-none-any.whl (11 kB)
Downloading platformdirs-4.2.0-py3-none-any.whl (17 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.8 filelock-3.13.4 platformdirs-4.2.0 virtualenv-20.25.3

C:\Users\Kundana G>python -m venv venv

C:\Users\Kundana G>
C:\Users\Kundana G>virtualenv venv
created virtual environment CPython3.12.3.final.0-64 in 5880ms
  creator CPython3Windows(dest=C:\Users\Kundana G\venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\Kundana G\AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\Kundana G>venv\Scripts\activate

C:\Users\Kundana G>()

(venv) C:\Users\Kundana G>pip install flask
```

Fig-4.1.5:Install Flask

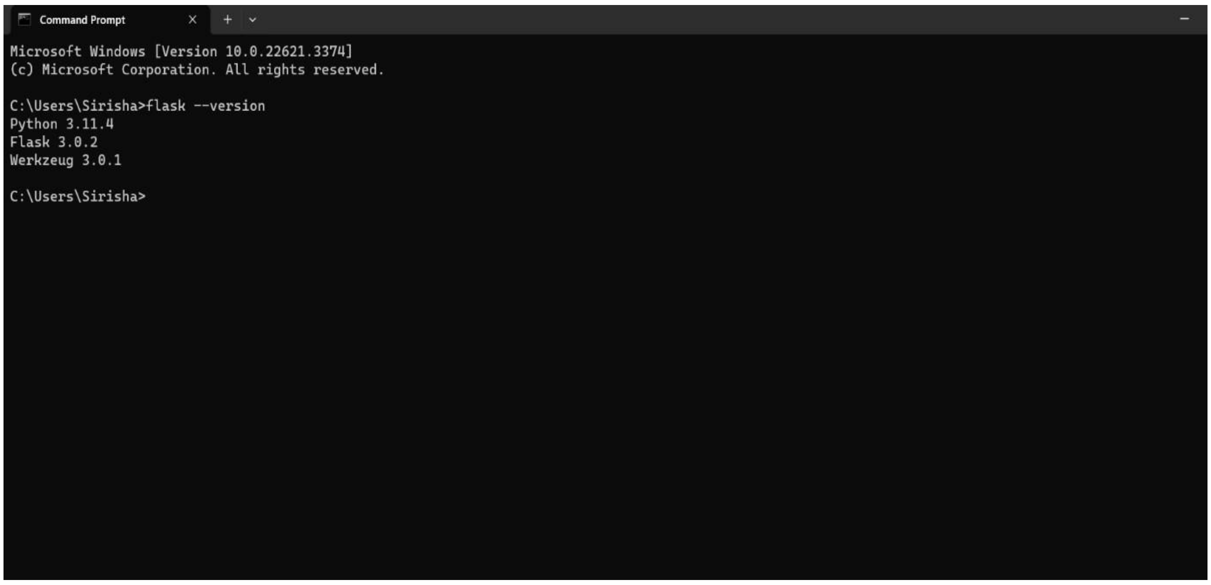


```
seeder FromAppData(download=False, pip=bundle, via=copy, app_data_dir=C:\Users\Kundana G\AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\Kundana G>venv\Scripts\activate

C:\Users\Kundana G>()

(venv) C:\Users\Kundana G>pip install flask
Collecting flask
  Downloading flask-3.0.3-py3-none-any.whl.metadata (3.2 kB)
Collecting Werkzeug>=3.0.0 (from flask)
  Downloading werkzeug-3.0.2-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from flask)
  Downloading Jinja2-3.1.3-py3-none-any.whl.metadata (3.3 kB)
Collecting itsdangerous>=2.1.2 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from flask)
  Downloading blinker-1.7.0-py3-none-any.whl.metadata (1.9 kB)
Collecting colorama (from click>=8.1.3->flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
  Downloading MarkupSafe-2.1.5-cp312-cp312-win_amd64.whl.metadata (3.1 kB)
Downloading flask-3.0.3-py3-none-any.whl (101 kB)
   ---------------------------------------- 101.7/101.7 kB 586.5 kB/s eta 0:00:00
Downloading blinker-1.7.0-py3-none-any.whl (13 kB)
Downloading click-8.1.7-py3-none-any.whl (97 kB)
   ---------------------------------------- 97.9/97.9 kB 1.9 MB/s eta 0:00:00
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading Jinja2-3.1.3-py3-none-any.whl (133 kB)
   ---------------------------------------- 133.2/133.2 kB 2.0 MB/s eta 0:00:00
Downloading werkzeug-3.0.2-py3-none-any.whl (226 kB)
   ---------------------------------------- 226.8/226.8 kB 7.0 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp312-cp312-win_amd64.whl (17 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, blinker, Werkzeug, Jinja2, click, flask
Successfully installed Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.2 blinker-1.7.0 click-8.1.7 colorama-0.4.6 flask-3.0.3 itsdangerous-2.2.0

(venv) C:\Users\Kundana G>
```

Fig-4.1.6:Downloading Flask

Step – 5: Verify Installation: After installation, you can verify that Flask is installed correctly by running the following command: flask --version

The Flask installed on your system.



```
Microsoft Windows [Version 10.0.22621.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sirisha>flask --version
Python 3.11.4
Flask 3.0.2
Werkzeug 3.0.1

C:\Users\Sirisha>
```

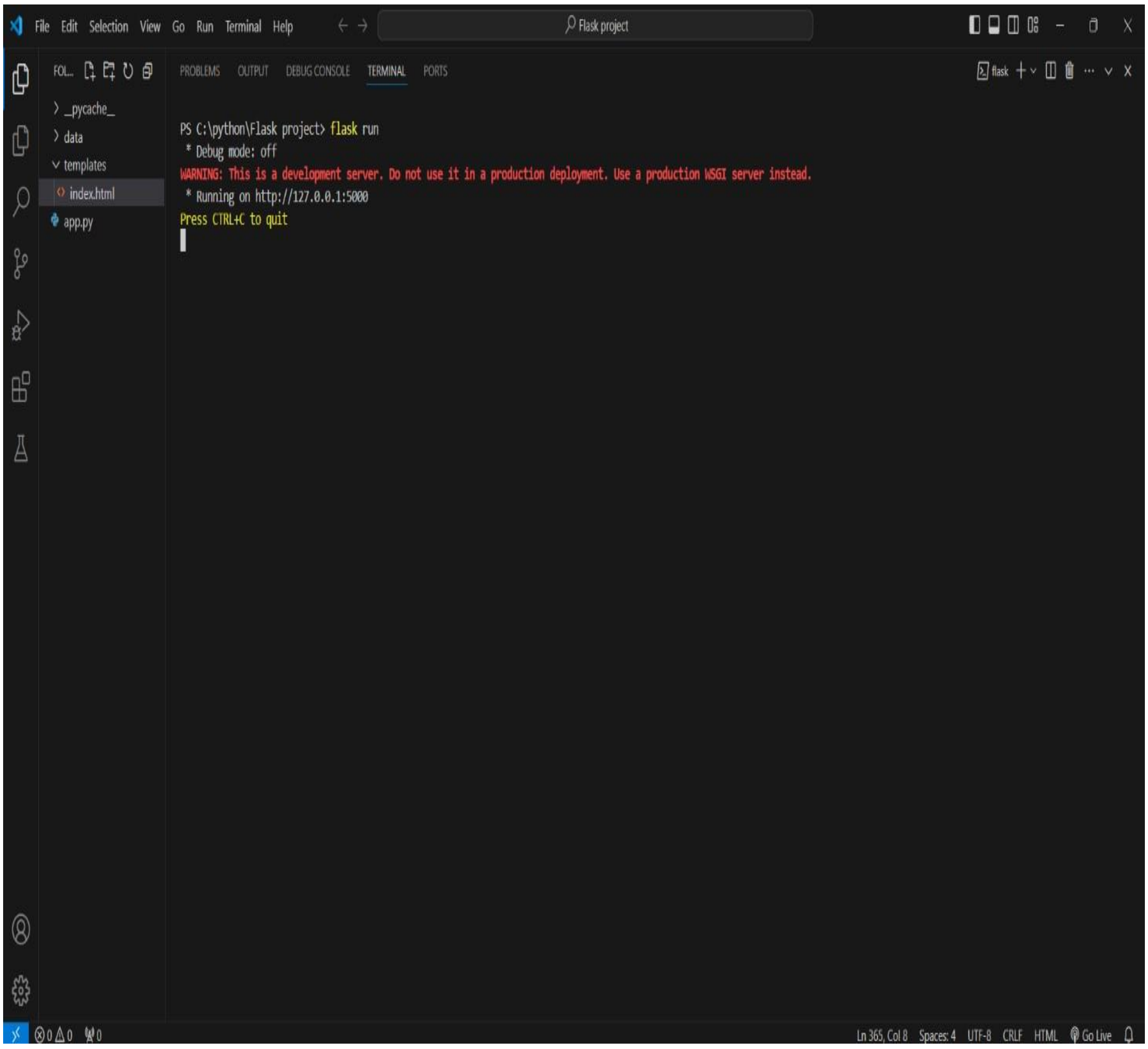
Fig-4.1.7:Flask Verification

Step – 6: Run the Flask Application

With the Flask application created, we can now run it using the following command:

```
flask run
```

This command will start the Flask development server, and you'll be able to access your application by navigating to http://127.0.0.1:5000 in your web browser.



```
PS C:\python\Flask project> flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Fig-4.1.8:Running Flask Application

DATA MODULES

Data modules refer to discrete units within a system that handle specific aspects of data management, such as gathering, processing, or storage, often modularized for scalability and maintainability. They encapsulate functionalities related to data operations to promote organization and reusability within software architecture.

Data Gathering:

The project sources its weather data from the National Atmospheric Research Laboratory (NARL), which offers a diverse range of weather parameters like surface temperature, rainfall, humidity, and wind speed. To ensure the forecasts are always up-to-date, data is retrieved at regular intervals. This data gathering process guarantees that users receive accurate and timely weather information for their selected locations. NARL's data is crucial for providing reliable weather forecasts through the application, enhancing its utility and user experience.

Data Processing:

- Data processing involves converting raw data into a more meaningful format. For weather forecasting, this might include steps like data cleaning, transformation, and analysis to derive insights.
- In the context of weather forecasting, the conversion of raw data into machine-readable form is a critical step in the data processing phase. Raw weather data, which includes measurements such as temperature, humidity, wind speed, and precipitation, is typically collected from various sources such as weather stations, satellites, and weather models. However, this raw data is often in a format that is not directly usable by computers or algorithms.
- The conversion process involves transforming the raw data into a format that can be easily processed and analyzed by machines. This typically involves converting the data into a structured format, such as a table or a database, where each data point is represented as a row with specific attributes or columns.
- For example, raw weather data collected from a weather station might be in the form of unstructured text or binary data. To make this data machine-readable, it needs to be parsed and converted into a structured format that can be easily manipulated and analyzed. This could involve extracting relevant information such as date, time, location, and weather parameters from the raw data and organizing it into a structured format.
- Once the raw data is converted into a machine-readable form, it can be processed further using various techniques such as data cleaning, transformation, and analysis to derive meaningful insights. These insights can then be used to generate weather forecasts, identify patterns and trends, and make informed decisions related to weather-related risks and planning.
- Overall, the conversion of raw weather data into machine-readable form is a crucial step in the data processing phase, as it lays the foundation for further analysis and interpretation of the data.

Data Analysis:

- In the context of weather forecasting, Python is a popular programming language used for data analysis due to its versatility and the availability of powerful libraries. When raw weather data is retrieved, Python libraries can be used to process and analyze this data to extract relevant information.
- One of the key libraries used for data analysis in Python is pandas. Pandas provides data structures and functions that make it easy to manipulate and analyze structured data, such as weather data. For example, pandas can be used to read raw weather data from a file or database into a pandas DataFrame, which is a two-dimensional data structure that can be easily manipulated and analyzed.
- Once the raw weather data is loaded into a pandas DataFrame, various analysis techniques can be applied to extract relevant information. For example, you can use pandas to calculate summary statistics such as mean, median, and standard deviation for temperature or rainfall data. Pandas also provides powerful filtering and grouping capabilities, allowing you to extract data for specific time periods, locations, or weather conditions.
- In addition to pandas, other Python libraries such as NumPy, matplotlib, and seaborn can be used for data analysis and visualization. NumPy provides support for mathematical operations on arrays, which is useful for performing calculations on weather data. Matplotlib and seaborn are used for creating visualizations such as line charts, scatter plots, and histograms, which can help to visualize patterns and trends in the weather data.
- Overall, Python libraries play a crucial role in the analysis of raw weather data, allowing you to extract relevant information and gain insights that can be used for weather forecasting and decision making.

Implementation of user interface:

- A responsive user interface is crucial for providing users with easy access to weather forecasts for their desired locations. In the context of weather forecasting applications, a responsive user interface refers to a design that adapts and functions well across a variety of devices and screen sizes, such as desktops, tablets, and smartphones. This ensures that users can access weather information anytime, anywhere, using their preferred device.
- To achieve this, the user interface should be designed with a focus on usability and accessibility. This includes features such as a clean and intuitive layout, easy-to-use navigation menus, and interactive elements that allow users to quickly access the information they need. For example, the interface could include a search bar or dropdown menu where users can select their desired location and date to view the weather forecast.
- In addition, the user interface should be designed to display weather information in a clear and concise manner. This might include using visual elements such as icons, graphs, and charts to represent weather data such as temperature, precipitation, and wind speed. The interface should also provide the option for users to customize their view, such as selecting which weather parameters they want to see or changing the time range of the forecast.

CHAPTER - 5

IMPLEMENTATION

Front End Code:

Sample.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>India Map</title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css"
/>
    <style>
      body {
        display: flex;
        flex-direction: column;
        height: 100vh;
        margin: 0;
        background-color: skyblue;
      }
      img {
        max-width: 43%;
        align-self: left;
      }
      .container,
      .time-container {
        display: flex;
        flex: 1;
      }
      input[type="button"],
      .time-slider {
        background-color: lightpink;
      }
      form {
        padding: 9%;
        flex: 1;
      }
      #select {
        font-size: 20px;
      }
      .box,
      .time-slider {
        width: 250px;
        padding: 5px;
      }
```

```
.button {
  padding: 10px 20px;
  font-size: 16px;
}
#map {
  height: 60%;
  flex: 2;
  padding-top: 10%;
}

.time-slider {
  width: 50%;
  margin: 20px 5% 20px auto;
  align-self: flex-end;
  height: 50px; /* Adjust height */
  border-radius: 0; /* Make it rectangular */
  overflow: hidden; /* Hide overflow */
}
.time-slider input[type="range"] {
  width: 100%; /* Fill the container */
  height: 100%; /* Fill the container */
  -webkit-appearance: none; /* Remove default styles */
  appearance: none; /* Remove default styles */
  background: lightpink; /* Background color */
  outline: none; /* Remove outline */
  border: none; /* Remove border */
  border-radius: 0; /* Round corners */
}
.time-slider input[type="range"]::-webkit-slider-thumb {
  -webkit-appearance: none; /* Remove default styles */
  appearance: none; /* Remove default styles */
  width: 20px; /* Width of the thumb */
  height: 20px; /* Height of the thumb */
  background: white; /* Thumb color */
  border-radius: 50%; /* Circle shape */
  cursor: pointer; /* Cursor on hover */
}
@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
  #map,
  .time-slider {
    width: 100%;
    height: 250px;
  }
  .time-slider {
    width: 90%;
```

```
          margin: 20px auto;
        }
      }
      .close {
        color: #aaa;
        float: right;
        font-size: 28px;
        font-weight: bold;
        cursor: pointer;
      }
      .close:hover,
      .close:focus {
        color: black;
        text-decoration: none;
      }
    </style>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  </head>
  <body>
    <img src="{{ url_for('static', filename='wcrglogo.jpg') }}" alt="Weather"
/>
    <div>
      <img id="imageDisplay" />
    </div>
    <div class="container">
      <form id="generateForm">
        <label for="initial" id="select"
          ><b>Select forecast initial date:</b></label
        ><br /><br />
        <!-- initialDate -->
        <input
          type="date"
          class="box"
          name="initial"
          id="initialDate"
        /><br /><br />
        <label for="forecastDates" id="select"
          ><b>Select forecast dates for 7 days:</b></label
        ><br /><br />
        <select name="forecastDates" id="forecastDates" class="box"></select
        ><br /><br />
        <label for="stateSelect" id="select"><b>Select State</b></label
        ><br /><br>
        <!-- stateName -->
        <select id="stateSelect" class="box">
          <!-- Dropdown filled by JavaScript -->
        </select>
        <br /><br>
        <label for="parameter" id="select"><b>Select parameter:</b></label
        ><br /><br />
        <select name="para" id="para" class="box">
          <option value="rainfall">Rainfall</option>
          <option value="temperature">Surface Temperature</option>
          <option value="humidity">Humidity</option>
          <option value="wind">Wind speed</option></select
        ><br /><br /><br>
<label for="select" id="select"><b>Select hour from the slider:</b></label>

          <!-- <input
          type="button"
          class="button"
          value="Submit"
          onclick="changeMarkerColor()"
        /> -->
        <div class="time-container">
          <input
            type="range"
            class="time-slider"
            min="0"
            max="24"
            value="0"
            id="timeSlider"
          />
        </div>
        <button type="submit" class="button" onclick="changeMarkerColor()">
          Submit
        </button>
      </form>
      <div id="map"></div>
    </div>

    <div id="myModal" class="modal">
      <div class="modal-content">
        <span class="close" onclick="closeModal()">&times;</span>
        <iframe
          id="modalContent"
          src=""
          style="width: 100%; height: 100%"
          frameborder="0"
        ></iframe>
      </div>
    </div>
    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <script>
      var map = L.map("map").setView([21.5937, 78.9629], 5);
      L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
        attribution:
          '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors',
      }).addTo(map);

      var markersData = [
        {name: "Andaman and Nicobar Islands", lat: 10.2188343994493,lng:
92.57713289999998,},
        {name: "Andhra Pradesh",lat: 15.924090500015737,lng:
80.18638089999999,},
{ name: "Arunachal Pradesh", lat: 28.093770199935786, lng: 94.5921326 },
{ name: "Assam", lat: 26.40738410012204, lng: 93.2551303 },
{ name: "Bihar", lat: 25.64408450019941, lng: 85.906508 },
{ name: "Chhattisgarh", lat: 21.66373590041017, lng: 81.8406351 },
{ name: "Delhi", lat: 28.62739279987712, lng: 77.1716954 },
{ name: "Goa", lat: 15.300454299942789, lng: 74.08551339999998 },
{ name: "Gujarat", lat: 22.385005100402285, lng: 71.745261 },
{ name: "Haryana", lat: 28.99999999983744, lng: 76 },
{name: "Himachal Pradesh",lat: 31.929235199611334,lng: 77.18284619999999,},
{ name: "Jammu and Kashmir", lat: 32.732998, lng: 74.864273 },
{ name: "Jharkhand", lat: 23.455980900363866, lng: 85.25573009999998 },
{ name: "Karnataka", lat: 14.520389599850693, lng: 75.7223521 },
{ name: "Kerala", lat: 10.352874399456772, lng: 76.5120396 },
{name: "Madhya Pradesh",lat: 23.814341900344257,lng: 77.53407189999999,},
{ name: "Maharashtra", lat: 18.90683560030784, lng: 75.6741579 },
{ name: "Manipur", lat: 24.720881800281017, lng: 93.9538465 },
{ name: "Meghalaya", lat: 25.53794320020953, lng: 91.2999102 },
{ name: "Mizoram", lat: 23.214616900375216, lng: 92.8687612 },
{ name: "Nagaland", lat: 26.16305560014758, lng: 94.5884911 },
{ name: "Odisha", lat: 20.543124100393143, lng: 84.6897321 },
{ name: "Punjab", lat: 30.929321099665902, lng: 75.50048410000001 },
{ name: "Puducherry", lat: 11.934056, lng: 79.830645 },
{ name: "Rajasthan", lat: 26.81057770007866, lng: 73.7684549 },
{ name: "Tamil Nadu", lat: 10.909433399492215, lng: 78.36653469999999 },
{ name: "Telangana", lat: 17.849591900220112, lng: 79.1151663 },
{ name: "Tripura", lat: 23.77508230034657, lng: 91.7025091 },
{ name: "Uttarakhand", lat: 30.04173759973603, lng: 79.08969099999999 },
{name: "West Bengal", lat: 22.996494800384152,lng: 87.68558819999998,}

                          ];

      var markers = {};
      var isRed = {};

      markersData.forEach(function (markerData) {
        var marker = L.marker([markerData.lat, markerData.lng])
          .addTo(map)
          .bindPopup(markerData.name);
        markers[markerData.name] = marker;
        isRed[markerData.name] = false;

        var option = document.createElement("option");
        option.value = markerData.name;
        option.text = markerData.name;
        document.getElementById("stateSelect").appendChild(option);
        // onclick for marker tag
        marker.on("click", function () {
          if (isRed[markerData.name]) {
            window.location.href = "/getimage";
            // var modal = document.getElementById("myModal");
            // var modalContent = document.getElementById("modalContent");
            // modal.style.display = "block";
            // modalContent.src = "/getimage";
          }
        });
      });

      $(document).ready(function () {
        $("#generateForm").submit(function (event) {
          event.preventDefault();
          var formData = {
            date2: $("#forecastDates").val(),
            state_name: $("#stateSelect").val(),
            hour: $("#timeSlider").val().padStart(2, "0"),
            para: $("#para").val(),
          };
          // Send AJAX POST request to the server
          $.ajax({
            type: "POST",
            url: "/",
            contentType: "application/json",
            data: JSON.stringify(formData),
            success: function (response) {
              // Handle success response if needed
              console.log(response);
            },
            error: function (xhr, status, error) {
              // Handle error response if needed
              console.error(error);
            },
          });
        });
      });
```

```
      function changeMarkerColor() {
        var selectedState = document.getElementById("stateSelect").value;

        Object.keys(markers).forEach(function (stateName) {
          var defaultIcon = new L.Icon.Default();
          markers[stateName].setIcon(defaultIcon);
          isRed[stateName] = false;
        });

        var redIcon = new L.Icon({
          iconUrl:
            "https://raw.githubusercontent.com/pointhi/leaflet-color-
markers/master/img/marker-icon-2x-red.png",
          shadowUrl:
            "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marke
r-shadow.png",
          iconSize: [25, 41],
          iconAnchor: [12, 41],
          popupAnchor: [1, -34],
          shadowSize: [41, 41],
        });
        markers[selectedState].setIcon(redIcon);
        isRed[selectedState] = true;
      }

     document.getElementById("initialDate").addEventListener("change",
function () {
    var initialDate = new Date(this.value);
    var forecastDatesSelect = document.getElementById("forecastDates");
    forecastDatesSelect.innerHTML = ""; // Clear previous options

    for (var i = 0; i < 7; i++) {
        var date = new Date(initialDate);
        date.setDate(initialDate.getDate() + i);

        // Formatting date to YYYY-MM-DD format
        var formattedDate = date.getFullYear() + '-' + (date.getMonth() +
1).toString().padStart(2, '0') + '-' + date.getDate().toString().padStart(2,
'0');

        var option = document.createElement("option");
        option.value = formattedDate;
        option.text = formattedDate;
        forecastDatesSelect.appendChild(option);
    }
});

      // JavaScript code for fetching and displaying images
      document
        .getElementById("timeSlider")
        .addEventListener("input", function () {
          var sliderValue = this.value;
          console.log("Slider value: " + sliderValue); // Replace this with
functionality to adjust the map based on time

          // Fetch image based on slider value
          fetch(`/images/${sliderValue}.jpg`)
            .then((response) => {
              if (!response.ok) {
                throw new Error("Image not found");
              }
              return response.blob();
            })
            .then((blob) => {
              const imageUrl = URL.createObjectURL(blob);
              document.getElementById("imageDisplay").src = imageUrl;
            })
            .catch((error) => {
              console.error(error);
            });
        });
    </script>
  </body>
</html>
```

This is an HTML page that includes JavaScript and CSS. It's designed to create a web interface for selecting parameters related to weather forecasts and displaying them on a map.

- HTML Structure: The HTML structure defines various elements such as forms, buttons, sliders, maps, and modals.
- CSS Styling: Defines the styling for different elements to create a visually appealing interface.
- JavaScript: Handles user interactions and makes AJAX requests to the server for data.
- Leaflet.js: Integrates Leaflet library for displaying interactive maps.
- jQuery: Utilizes jQuery library for DOM manipulation and AJAX request.

getimages.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Dynamic Image</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        margin: 0;
        padding: 0;
        background-color: #f2f2f2;
      }

      .container {
        max-width: 800px;
        margin: 0 auto;
        padding: 20px;
      }

      h1 {
        text-align: center;
      }

      .time-container {
        text-align: center;
        margin-bottom: 20px;
      }

      .time-slider {
        width: 80%;
      }

      #imageDisplay {
        display: block;
        margin: 0 auto;
        max-width: 100%;
        height: auto;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h1>Image Display</h1>
      <div class="time-container">
        <input
          type="range"
          class="time-slider"
          min="0"
          max="23"
          value="0"
          id="timeSlider"
        />
      </div>

      <img
        src="{{ url_for('static', filename='images/' + para +'/' + date2 + '_'
+ hour + '.' + state + '.png') }}"
        alt="Image individual"
        id="imageDisplay"
      />
    </div>

    <script>
      var timeSlider = document.getElementById("timeSlider");
      var imageDisplay = document.getElementById("imageDisplay");

      // Add event listener to the range input
      timeSlider.addEventListener("input", function () {
        var hour =
          timeSlider.value < 10 ? "0" + timeSlider.value : timeSlider.value;
        imageDisplay.src =
          "{{ url_for('static', filename='images/') }}" +
          "{{ para }}" +
          "/" +
          "{{ date2 }}" +
          "_" +
          hour +
          "." +
          "{{ state }}" +
          ".png";
      });
    </script>
  </body>
</html>
```

This HTML file is another part of the web application, specifically for displaying dynamic images based on user-selected parameters.

- HTML Structure: Contains a range input for selecting hours and an image tag to display the corresponding image.
- JavaScript: Updates the image source dynamically based on the selected hour.

Backend Code:

app.py

```python
import json
from flask import Flask, jsonify, request, render_template, redirect, url_for

app = Flask(__name__)

received_data = {}  # Initialize an empty dictionary to store received data


@app.route('/', methods=['GET', 'POST'])
def get_data():
    global received_data  # Use global keyword to access the global variable
    if request.method == 'POST':
        if request.is_json:
            received_data = request.json  # Update global variable with
received data
            print("Received data '/' :", received_data)
            return jsonify({'data': received_data}), 200
        else:
            print("Invalid JSON data in request.")
            return jsonify({'error': 'Invalid JSON data in request.'}), 400
    else:
        return render_template("sample.html")


@app.route('/getimage', methods=['GET'])
def index():
    global received_data  # Use global keyword to access the global variable
    date2=received_data.get('date2')
    hour=received_data.get('hour')
    state_name = received_data.get('state_name', '')
    state = state_name.split()[0]
    para = received_data.get('para', '')


    image_filename =
f"{received_data.get('date2')}_{received_data.get('hour')}.{state}.png"
    image_url = url_for('static', filename=f'images/{image_filename}')

    print("Received data '/getimage' :", image_url)
    return render_template('getImages.html',
image_url=image_url,date2=date2,hour=hour,state=state,para=para)
if __name__ == '__main__':
app.run(debug=True)
```

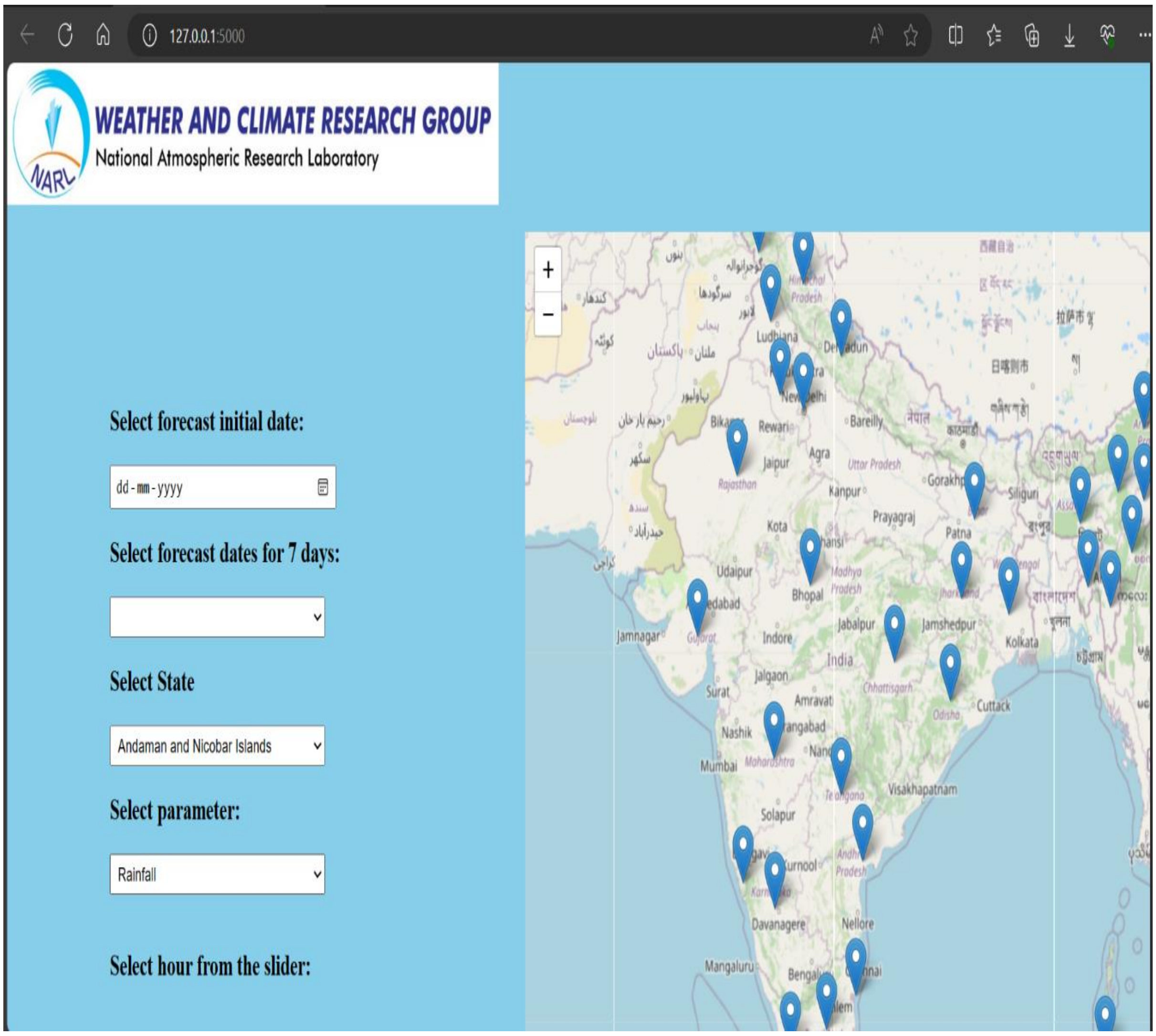
This is a Python Flask application that serves as the backend for the web application.

- ✓ Flask Routes: Defines two routes:
- ✓ Handles both GET and POST requests. It renders a template (sample.html) for the main page and handles POST requests to update received data.
- ✓ getimage: Handles GET requests to retrieve images based on the received data and renders a template (getImages.html) for displaying the image.
- ✓ Main Function: Runs the Flask application.

CHAPTER- 6

RESULT AND ANALYSIS

This project is about storing the state wise past weather data for few weather parameters surface temperature, rainfall, humidity, wind speed in the form of images .The stored data is for 24 hours with the variations in weather conditions.


127.0.0.1:5000
WEATHER AND CLIMATE RESEARCH GROUP
National Atmospheric Research Laboratory
NARL
+
−

Select forecast initial date:

dd-mm-yyyy

Select forecast dates for 7 days:

Select State

Andaman and Nicobar Islands

Select parameter:

Rainfall

Select hour from the slider:

Fig: This is the frontend view of the weather forecasting application.

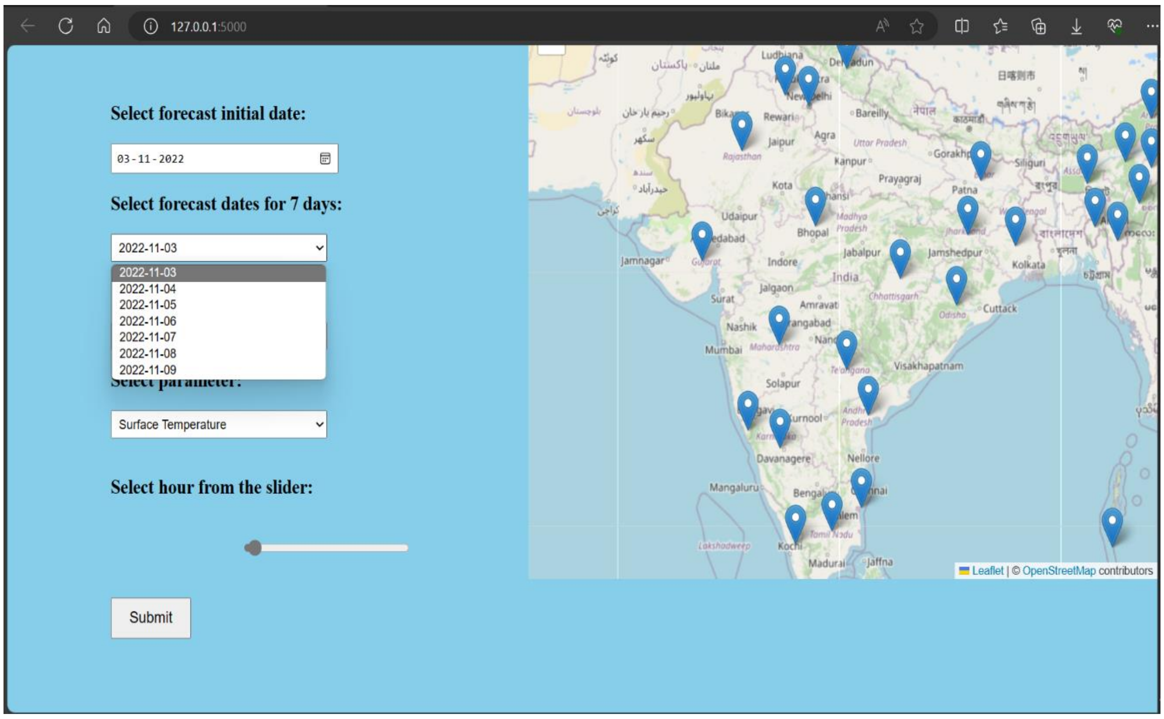


Fig: Select the initial date

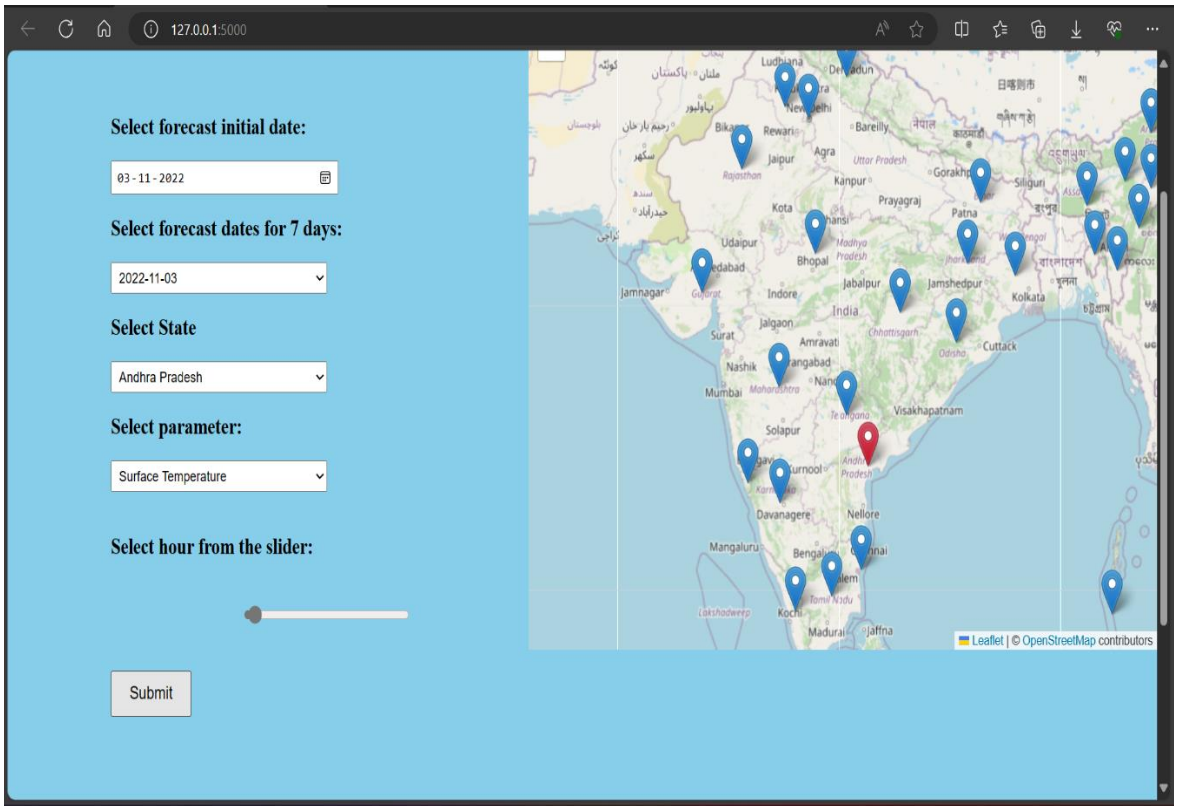

127.0.0.1:5000
Select forecast initial date:
03-11-2022
Select forecast dates for 7 days:
2022-11-03
Select State
Andhra Pradesh
Select parameter:
Surface Temperature
Select hour from the slider:
Submit
Leaflet | © OpenStreetMap contributors

Fig: Select all the fields as per the user requirement and submit this results in highlighting the state.

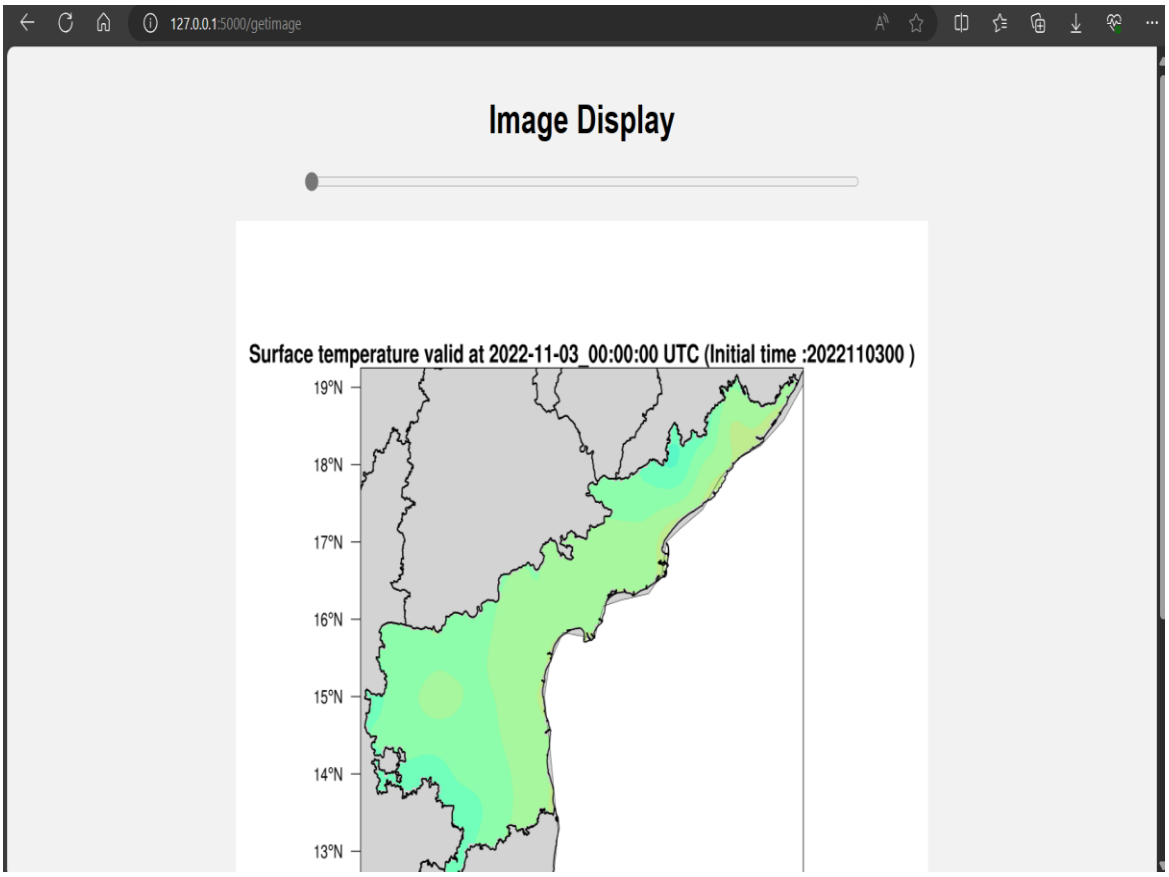

127.0.0.1:5000/getimage
Image Display
Surface temperature valid at 2022-11-03_00:00:00 UTC (Initial time :2022110300)
19°N
18°N
17°N
16°N
15°N
14°N
13°N

Fig: On clicking the red indicator it views all the data which is changeable by moving the time slider

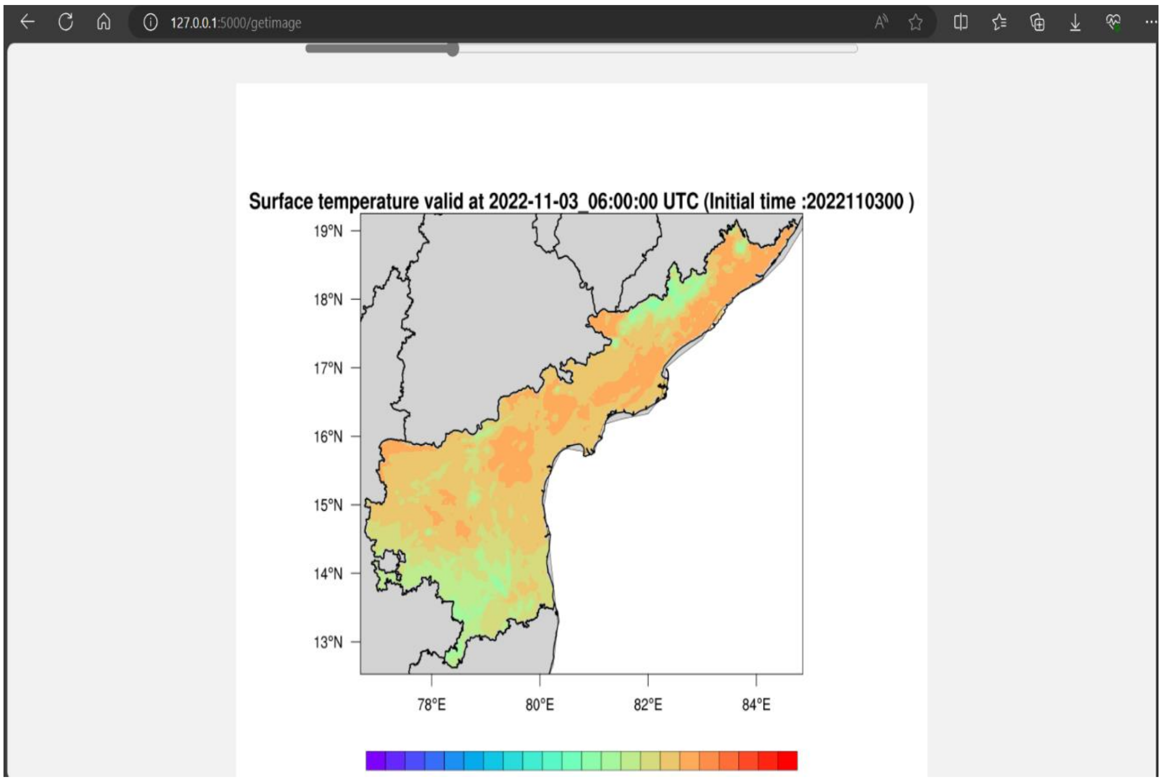

127.0.0.1:5000/getimage
Surface temperature valid at 2022-11-03_06:00:00 UTC (Initial time :2022110300)
19°N
18°N
17°N
16°N
15°N
14°N
13°N
78°E
80°E
82°E
84°E

Fig: Here the data at 6th hour is viewable by moving the time slider

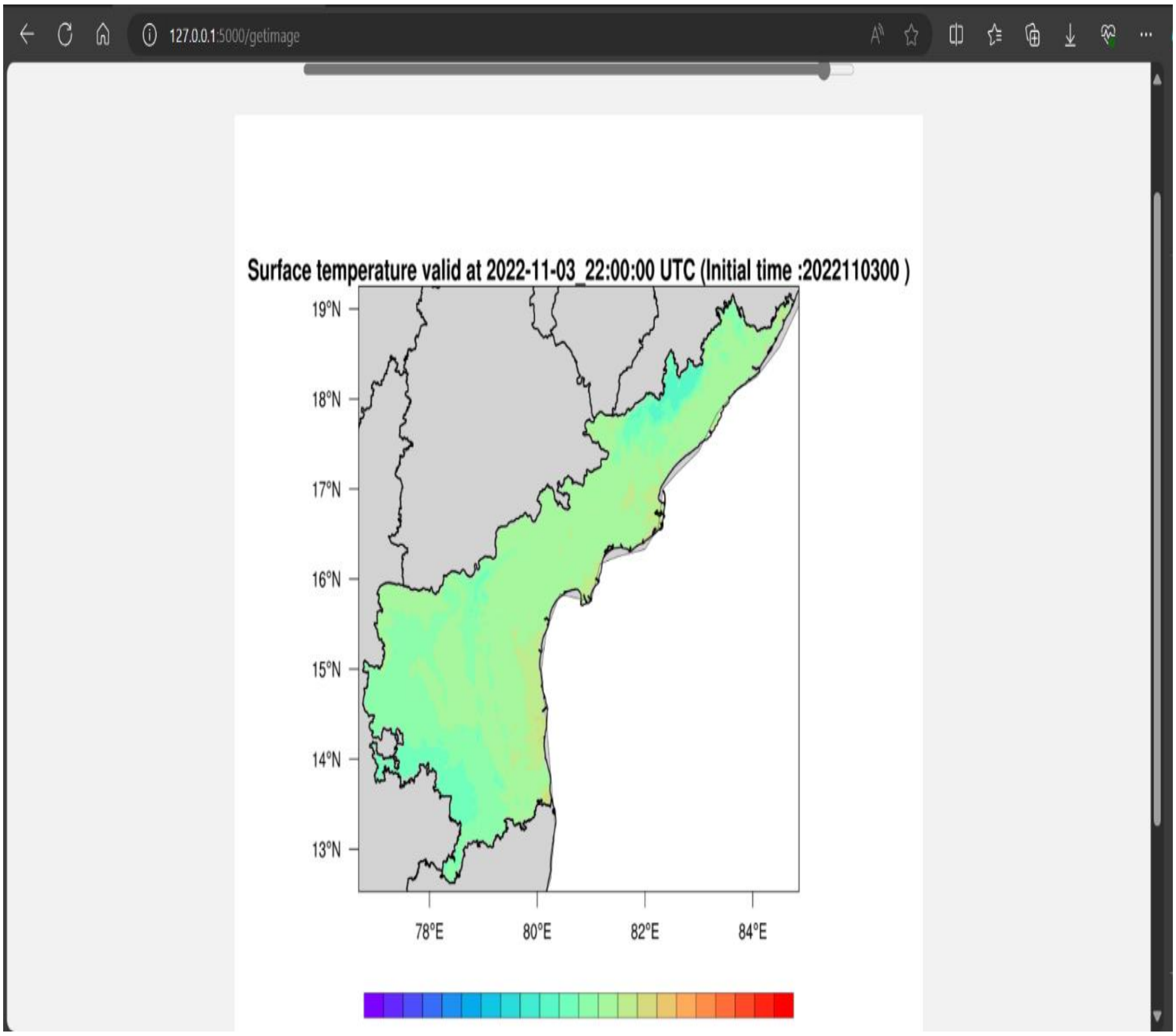

127.0.0.1:5000/getimage
Surface temperature valid at 2022-11-03_22:00:00 UTC (Initial time :2022110300)
19°N
18°N
17°N
16°N
15°N
14°N
13°N
78°E
80°E
82°E
84°E

Fig: This image displays the data for 22nd hour

CONCLUSION

Conclusion

The project showcases the potential of full stack development in addressing real-world challenges such as weather forecasting, and serves as a valuable resource for individuals, businesses, and organizations seeking access to accurate weather information. This project demonstrates the design and implementation of an affordable mini weather monitoring system that ensures flexibility, portability, scalability and user friendly operations which can provide data of some weather variables including temperature , humidity and rainfall.

REFERENCES

- Muthulakshmi, Priya, S. B. A survey on weather forecasting to predict rainfall using big data analytics.
- Smith, A., et al. (2020). "Advances in Location-based Weather Forecasting: A Machine Learning Perspective." *Journal of Weather Prediction*, 123-135.
- Brown, K., & Jones, L. (2020). "Convolutional Neural Networks for Image Analysis in Location-based Weather Forecasting." *Neural Computing and Applications*, 145-158.
- Wang, S., et al. (2020). "Deep Learning Techniques for Location-based Weather Prediction." IEEE Transactions on Geoscience and Remote Sensing, 28(4), 567-580.
- Johnson, B., & Lee, S. (2021). “Relationship modelling for Location-based Weather Forecasting." International Journal of Computational Meteorology and Climatology, 27(3), 210-224.
- Liu, Z., & Gupta, M. (2021). "Probabilistic Graphical Models for Location-based Weather Forecasting." IEEE Transactions on Geoscience and Remote Sensing, 25(3), 178-191.
- Patel, S., & Sharma, R. (2022). "Deep Learning Approaches for Location-Based Weather Forecasting." Advances in Meteorological Research, 20(2), 189-202.
- Clark, D., & White, R. (2022). “Enhancing Location-Based Weather Predictions with IoT Sensor Networks.”