**PROFESSIONAL INDUSTRIAL TRAINING**

**SUBMITTED TO**

**Department of Computer Science**

**And Engineering**

**Model Institute of Engineering and Technology (Autonomous)**

**Jammu, India**

**/ INTERNSHIP REPORT**

**ON**

“A Real time chat Application”

## AT

**AROOTLE PVT. LTD.**

**JAIPUR RAJASTHAN**

## AN INDUSTRY INTERNSHIP REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

**FOR THE AWARD OF DEGREE OF**

# BACHELOR OF ENGINEERING

**In**

# Department of Computer Science and Engineering

**SUBMITTED BY**

ADITYA PARTAP

Roll Number: 2020A1R113



**CANDIDATES’ DECLARATION**

I, **Aditya Partap , Roll Number (2020A1R113)** hereby declare that the work which is being presented in the Industry Internship Report entitled, “**A real time chat application”** in partial fulfillment of requirement for the award of degree of B.E. CSE and submitted in the Department of Computer Science And Engineering, Model Institute of Engineering and Technology (Autonomous), Jammu is an authentic record of my own work carried by me at “Arootle PVT. LTD, JAIPUR” under the supervision and mentorship of **Mr.** **Sarjeet Rao**( CEO of AROOTLE PVT. LTD). The matter presented in this report has not been submitted in this or any other University / Institute for the award of B.E. Degree.

*Signature of the Student Dated*:

## Aditya Partap 3RD OCT 2022

**2020A1R113**

**INTERNSHIP CERTIFICATE**

**Department Of Computer Science And Engineering**

**Model Institute of Engineering and Technology (Autonomous)**

**Kot Bhalwal, Jammu, India**

***(NAAC “A” Grade Accredited)***

**Ref. No.: 2020A1R113 Date:03RD OCT 2022**

# CERTIFICATE

Certified that this Industry Internship Report entitled **“Real time chat Application”** is the bonafide work of “**Aditya Partap , Roll No. 2020A1R113, of 5th Semester, Department of Computer Science And Engineering, Model Institute of Engineering and Technology (Autonomous), Jammu”,** who carried out the Industry Internship at **“AROOTLE PVT. LTD. JAMMU”** work under my mentorship during 20th July,2022 to 2nd September , 2022.

## (Dr. Swati Goel)

**Mentor-Internal Supervisor**

**Assistant Professor**

## Department of Computer Science and Engineering, MIET

*This is to certify that the above statement is correct to the best of my knowledge.*

**INTERNSHIP CERTIFICATE**



**ACKNOWLEDGEMENTS** 

This Summer internship opportunity was a great chance for learning and professional development. I am grateful for having a chance to meet so many wonderful people and professionals who led me through this internship period.

It is my pleasure to pay my heartfelt gratitude to Mr. Sarjeet Rao, CEO of AROOTLE PVT. LTD who have guided me through the course of this Internship.

I must record my deep sense of gratitude to Prof. (Dr.) Ankur Gupta (Director, MIET) and Prof. (Dr.) Ashok Kumar (Dean Academics & HOD CSE, MIET) for their guidance, constant inspiration and encouragement, and for their keen involvement throughout the course of present work.

Gratitude and thanks, although mean a very small thing to convey my thanks to my parents who have always given me a parental source of love, motivation and strength right from the journey of my life.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the teachers who in spite of being extraordinarily busy with their duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

I express my sincere gratitude to Arootle Pvt. Ltd Jaipur Rajasthan and Model Institute of Engineering and Technology (Autonomous), Jammu for giving me the opportunity.

**Aditya Partap**

**2020A1R113**

**ABSTRACT**

Weather forecasting is the application of science and technology to predict

the state of the atmosphere for a given location.Ancient weather forecasting

methods usually relied on observed patterns of events, also termed pattern

recognition. For example, it might be observed that if the sunset was particularly

red, the following day often brought fair weather.However, not all of these

predictions prove reliable.

Weather forecasting is the application of science and technology to predict

the state of the atmosphere for a given location.Ancient weather forecasting

methods usually relied on observed patterns of events, also termed pattern

recognition. For example, it might be observed that if the sunset was particularly

red, the following day often brought fair weather.However, not all of these

predictions prove reliable.

Weather forecasting is the application of science and technology to predict

the state of the atmosphere for a given location.Ancient weather forecasting

methods usually relied on observed patterns of events, also termed pattern

recognition. For example, it might be observed that if the sunset was particularly

red, the following day often brought fair weather.However, not all of these

predictions prove reliable.

Weather forecasting is the application of science and technology to predict

the state of the atmosphere for a given location.Ancient weather forecasting

methods usually relied on observed patterns of events, also termed pattern

recognition. For example, it might be observed that if the sunset was particularly

red, the following day often brought fair weather.However, not all of these

predictions prove reliable.

Weather forecasting is the application of science and technology to predict the state of atmosphere for a given location. Ancient Weather forecasting methods usually relied on observed patterns of events, also termed pattern recognition . for example, it might be observed that if the sunset was particularly red, the following day often brought fair weather. However, not all of these predictions prove reliable.

Here this system will predict weather based on parameters such as temperature, humidity and wind. User will enter current temperature; humidity and wind, System will take this parameter and will predict weather (rainfall in inches) from previous data in database (dataset). The role of the admin is to add plou Nous weather data in database, so that system will calculate weather (estimated rainfall in inches) based on these data. Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

|  |  |  |
| --- | --- | --- |
| ￼ | **CONTENTS** |  |
|  | **Page No.** |
| **CANDIDATES’ DECLARATION** | | **II** |
| **INTERNSHIP CERTIFICATE** | | **III** |
| **CERTIFICATE** | | **IV** |
| **ACKNOWLEDGEMENTS** | | **V** |
| **ABSTRACT** | | **VI** |
| **CONTENTS** | | **VII** |
| **LIST OF FIGURES** | | **XI** |
| **LIST OF TABLES** | | **XIII** |
|  | |  |
| **CHAPTER-1** | **INTRODUCTION TO WEB** | **14** |
|  | 1.1 INTRODUCTION | 14 |
|  | 1.2 HISTORY OF WEB | 14 |
|  | 1.2.1 Web browser | 16 |
|  | 1.2.2 Website | 16 |
|  | 1.2.3 Webpage | 16 |
|  | 1.3 CLIENT SERVER ARCHITECTURE | 16 |
|  |  |  |
| **CHAPTER 2** | **HTML** | **18** |
|  | 2.1 INTRODUCTION TO HTML | 18 |
|  | 2.2 HISTORY OF HTML | 18 |
|  | 2.3 HTML TAGS | 19 |
|  | 2.3.1 Web Browsers | 19 |
|  | 2.3.2 HTML Editors | 20 |
|  | 2.3.3 HTML Documents | 22 |
|  | 2.3.4 HTML Headings | 22 |
|  | 2.3.5 HTML Paragraphs | 23 |
|  | 2.3.6 HTML Links | 23 |
|  | 2.3.7 HTML Images | 23 |
|  | 2.3.8 HTML Buttons | 23 |
|  | 2.3.9 HTML Lists | 24 |
|  | 2.3.10 HTML Elements | 24 |
|  | 2.4 HTML TABLES | 24 |
|  | 2.4.1 HTML Table - Adding a Border | 26 |
|  | 2.4.2 HTML Table - Collapsed Borders | 26 |
|  | 2.4.3 HTML Table - Adding Cell Padding | 26 |
|  | 2.4.4 HTML Table - Left-align Headings | 27 |
|  | 2.4.5 HTML Table - Adding Border Spacing | 27 |
|  | 2.5 HTML FORMS | 27 |
|  | 2.5.1 Text Input | 28 |
|  | 2.5.2 Radio Button Input | 28 |
|  | 2.5.3 The Submit Button | 29 |
|  |  |  |
| **CHAPTER 3** | **CSS** | **30** |
|  | 3.1 INTRODUCTION | 30 |
|  | 3.2 HISTORY OF CSS | 30 |
|  | 3.3 CSS SYNTAX | 31 |
|  | 3.4 TYPES OF CSS | 31 |
|  | 3.4.1 Internal Style Sheet | 32 |
|  | 3.4.2 External Style Sheetg | 32 |
|  | 3.4.3 Inline Styles | 33 |
|  | 3.5 CSS COLORS | 33 |
|  | 3.5.1 CSS colors-hex codes | 33 |
|  | 3.5.2 CSS Background | 34 |
|  | 3.5.3 CSS Borders | 34 |
|  | 3.5.4 CSS TEXT | 35 |
|  | 3.5.5 Font Family | 35 |
|  | 3.5.6 CSS Tables | 36 |
|  |  |  |
| **CHAPTER 4** | **INTRODUCTION TO PYTHON** | **37** |
|  | 4.1 INTRODUCTION | 37 |
|  | 4.2 HISTORY OF PYTHON | 37 |
|  | 4.2.1 Why Python | 39 |
|  | 4.2.2 Application | 39 |
|  | 4.2.3 Features | 42 |
|  |  |  |
| **CHAPTER 5** | **BASICS OF PYTHON** | **45** |
|  | 5.1 LIST | 45 |
|  | 5.2 TUPLE | 47 |
|  | 5.3 SETS | 48 |
|  | 5.4 DICTIONARY | 49 |
|  | 5.5 FUNCTIONS | 50 |
|  | 5.6 INHERITANCE | 51 |

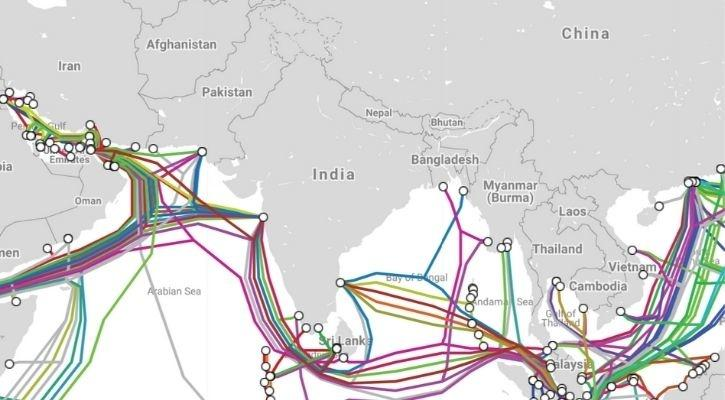
|  |  |  |
| --- | --- | --- |
| **CHAPTER 6** | **SQL** | **55** |
|  | 6.1 INTRODUCTION | 55 |
|  | 6.1.1 DDL: Data Definition Language | 55 |
|  | 6.1.2 DML: Data Manipulation Language | 55 |
|  | 6.1.3 TCL: Transaction Control Language | 56 |
|  | 6.1.4 DCL: Data Control Language | 56 |
|  | 6.2 SQL LANGUAGE ELEMENTS | 57 |
|  | 6.2.1 Retrieving Tables | 58 |
|  | 6.2.2 Selecting Columns from a Table | 58 |
|  | 6.2.3 Outputting Data Using a Constraint | 59 |
|  | 6.2.4 Sorted Data Using ‘Order By’ | 59 |
|  | 6.2.5 Outputting Sorted Data Using ‘Group By’ | 59 |
|  | 6.2.6 Data Manipulation Using COUNT | 60 |
|  | 6.2.7 Data Manipulation Using SUM | 60 |
|  | 6.2.8 Data Manipulation Using AVG | 60 |
|  | 6.2.9 Query for Listing all Views | 60 |
|  | 2.2.10 Query for Creating a View | 60 |
|  | 6.2.11 Advantages of SQL | 61 |
|  | 6.2.12 Disadvantages of SQL | 61 |
|  |  |  |
| **CHAPTER 7** | **DJANGO** | **62** |
|  | 7.1 INTRODUCTION | 62 |
|  | 7.2 INSTALLATION | 62 |
|  | 7.3 CREATE PROJECT | 68 |
|  | 7.4 CREATE APPLICATION | 69 |
|  | 7.5 ADMIN INTERFACE | 70 |
|  | 7.6 MODELS | 70 |
|  | 7.7 GENERIC VIEWS | 71 |
|  |  |  |
| **CHAPTER 8** | **MERN** | **73** |
|  | 8.1 INTRODUCTION | 73 |
|  | 8.2 MongoDB | 73 |
|  | 8.3 Express.js | 74 |
|  | 8.4 React | 75 |
|  | 8.5 Node.js | 76 |
| **CHAPTER 9** | **WEATHER APP** | **77** |
|  | 9.1 INTRODUCTION | 77 |
|  | 9.2 PROBLEM STATEMENT | 77 |
|  | 9.3 OBJECTIVE | 78 |
|  | 9.4 SCOPE | 78 |
|  | 9.5 SYSTEM REQUIREMENT | 79 |
|  | 9.6 WORK FLOW | 79 |
|  | 9.7 SNAPSHOTS | 80 |
|  | 9.8 CONCLUSION | 84 |
|  |  |  |
| **REFERENCES** |  | **85** |

|  |  |  |
| --- | --- | --- |
| **Fig No.** | **Figure** | **Page No.** |
| Figure 1.1 | Global map of Web | 14 |
| Figure 2.1 | HTML Code | 21 |
| Figure2.2 | Saving HTML Code | 21 |
| Figure 2.3 | Output of HTML Code | 22 |
| Figure 3.1 | CSS SYNTAX | 31 |
| Figure 4.1 | Python LOGO | 37 |
| Figure 4.2 | Python History | 38 |
| Figure 4.3 | Web Code | 40 |
| Figure 4.4 | Game Library | 40 |
| Figure 4.5 | AI & ML | 41 |
| Figure 4.6 | Data Analysis | 41 |
| Figure 4.7 | GUI | 42 |
| Figure 4.8 | Media | 42 |
| Figure 5.1 | Slicing | 47 |
| Figure 5.2 | Tuples | 48 |
| Figure 5.3 | SET | 48 |
| Figure 5.4 | Dictionary | 49 |
| Figure 5.5 | Functions | 50 |
| Figure 5.6 | Single Inheritance | 52 |
| Figure 5.7 | Multiple Inheritance | 52 |
| Figure 5.8 | Multilevel Inheritance | 53 |
| Figure 5.9 | Hierarchical Inheritance | 53 |
| Figure 5.10 | Hybrid Inheritance | 54 |
| Figure 7.1 | Step 1 Installing pip | 63 |
| Figure 7.2 | Step 2 Installing virtual env | 63 |
| Figure 7.3 | Step 3 Set virtual env | 64 |
| Figure 7.4 | Step 4 Change directory | 64 |
| Figure 7.5 | Step 5 Activate script | 65 |
| Figure 7.6 | Step 6 Installing django | 65 |
| Figure 7.7 | Step 7 Set directory | 66 |
| Figure 7.8 | Step 8 Start project | 66 |
|  |  |  |
|  |  |  |
|  |  |  |
| Figure 7.9 | Step 9 Change directory | 67 |
| Figure 7.10 | Step 10 Start server | 67 |
| Figure 7.11 | Step 11 Start URL | 68 |
| Figure 8.1 | MERN | 73 |
| Figure 9.1 | Flow Diagram | 79 |
| Figure 9.2 | HTML CODE | 80 |
| Figure 9.3 | CSS CODE | 80 |
| Figure 9.4 | PYTHON CODE | 81 |
| Figure 9.5 | PYTHON FILES | 81 |
| Figure 9.6 | CSS FILES | 81 |
| Figure 9.7 | FILES | 81 |
| Figure 9.8 | EXECUTION | 82 |
| Figure 9.9 | Page | 82 |
| Figure 9.10 | 8000 / page by Django | 83 |
| Figure 9.11 | Home Page | 83 |
| Figure 9.12 | Weather page | 84 |

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table** | **Page No.** |
| Table 2.1 | Basic HTML | 26 |
| Table 2.2 | HTML Forms | 27 |
| Table 5.1 | List Method | 45 |
| Table 5.2 | Built-in Functions | 46 |
| Table 6.1 | Data Definition Language | 55 |
| Table 6.2 | Data Manipulation Language | 56 |
| Table 6.3 | Transaction Control Language | 56 |
| Table 6.4 | Data Control Language | 56 |

**CHAPTER 1 INTRODUCTION TO WEB**

# INTRODUCTION

Figure 1.1 Global Map of Web

The World Wide Web (WWW), also called the Web, is an [information space](https://en.wikipedia.org/wiki/Information_space) where documents

and other [web resources](https://en.wikipedia.org/wiki/Web_resource) are identified by [Uniform Resource Locators](https://en.wikipedia.org/wiki/Uniform_Resource_Locator) (URLs), interlinked by [hypertext](https://en.wikipedia.org/wiki/Hypertext) links, and accessible via the [Internet](https://en.wikipedia.org/wiki/Internet). English scientist [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee) invented the World Wide Web in 1989. He wrote the first [web browser](https://en.wikipedia.org/wiki/Web_browser) in 1990 while employed at [CERN](https://en.wikipedia.org/wiki/CERN) in Switzerland.The browser was released outside CERN in 1991, first to other research institutions starting in January 1991 and to the general public on the Internet in August 1991.

# HISTORY OF WEB

Sir Tim Berners-Lee is a British computer scientist. He was born in London, and his parents were early computer scientists, working on one of the earliest computers.

Growing up, Sir Tim was interested in trains and had a model railway in his bedroom. He [recalls](http://www.w3.org/People/Berners-Lee/Kids.html):

“I made some electronic gadgets to control the trains. Then I ended up getting more interested in electronics than trains. Later on, when I was in college I made a computer out of an old television set.”

After graduating from Oxford University, Berners-Lee became a software engineer at [CERN](http://home.web.cern.ch/), the large particle physics laboratory near Geneva, Switzerland. Scientists come from all over the world to use its accelerators, but Sir Tim noticed that they were having difficulty sharing information.

“In those days, there was different information on different computers, but you had to log on to different computers to get at it. Also, sometimes you had to learn a different program on each computer. Often it was just easier to go and ask people when they were having coffee…”, [Tim says](http://www.w3.org/People/Berners-Lee/Kids.html).

Tim thought he saw a way to solve this problem – one that he could see could also have much broader applications. Already, millions of computers were being connected together through the fast-developing [internet](http://www.internetsociety.org/internet/what-internet) and Berners-Lee realized they could share information by exploiting an emerging technology called hypertext.

In March 1989, Tim laid out his vision for what would become the web in a document called “[Information Management: A Proposal](http://info.cern.ch/Proposal.html)”. Believe it or not, Tim’s initial proposal was not immediately accepted. In fact, his boss at the time, [Mike Sendall](http://bullarchive.web.cern.ch/bullarchive/9930/art2/Text_E.html), noted the words “Vague but exciting” on the cover. The web was never an official CERN project, but Mike managed to give Tim time to work on it in September 1990. He began work using a [NeXT computer,](http://en.wikipedia.org/wiki/NeXT_Computer) one of Steve Jobs’ early products.

By October of 1990, Tim had written the three fundamental technologies that remain the foundation of today’s web (and which you may have seen appear on parts of your web browser):

* HTML: HyperText Markup Language. The markup (formatting) language for the web.
* URI: Uniform Resource Identifier. A kind of “address” that is unique and used to identify each resource on the web. It is also commonly called a URL.
* HTTP: Hypertext Transfer Protocol. Allows for the retrieval of linked resources from across the web.

Tim also wrote the first web page editor/browser (“WorldWideWeb.app”) and the first web server (“httpd“). By the end of 1990, the first web page was served on the open internet, and in 1991, people outside of CERN were invited to join this new web community.

* + 1. **Web browser**

A web browser (commonly referred to as a browser) is a [software application](https://en.wikipedia.org/wiki/Software_application) for accessing information on the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web). Each individual [web page](https://en.wikipedia.org/wiki/Web_page), image, and video is identified by a distinct [URL,](https://en.wikipedia.org/wiki/URL) enabling browsers to retrieve and display them on the [user's](https://en.wikipedia.org/wiki/User_(computing)) device.

A web browser is not the same thing as a [search engine](https://en.wikipedia.org/wiki/Search_engine), though the two are often confused. For a user, a search engine is just a [website](https://en.wikipedia.org/wiki/Website), such as [google.com](https://en.wikipedia.org/wiki/Google_Search), that stores searchable data about other websites. But in order to connect to and display websites on their device, a user needs to have a web browser installed.

The most popular web browsers are [Chrome](https://en.wikipedia.org/wiki/Google_Chrome), [Firefox](https://en.wikipedia.org/wiki/Firefox), [Safari](https://en.wikipedia.org/wiki/Safari_(web_browser)), [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer), and [Edge](https://en.wikipedia.org/wiki/Microsoft_Edge).

## Website

A website is a collection of related [web pages](https://en.wikipedia.org/wiki/Web_page), including [multimedia](https://en.wikipedia.org/wiki/Multimedia) content, typically identified with a common [domain name](https://en.wikipedia.org/wiki/Domain_name), and published on at least one [web server](https://en.wikipedia.org/wiki/Web_server). Notable examples are [wikipedia.org,](https://en.wikipedia.org/wiki/Wikipedia) [google.com](https://en.wikipedia.org/wiki/Google), and [amazon.com](https://en.wikipedia.org/wiki/Amazon_(company)). Today roughly 380 new websites are created every minute across the World. A website may be accessible via a public [InternetProtocol](https://en.wikipedia.org/wiki/Internet_Protocol) (IP) network, such as the [Internet,](https://en.wikipedia.org/wiki/Internet) or a private [local area](https://en.wikipedia.org/wiki/Local_area_network) [network](https://en.wikipedia.org/wiki/Local_area_network) (LAN), by referencing a [uniform resource locator](https://en.wikipedia.org/wiki/URL) (URL) that identifies the site.

## Webpage

A web page or webpage is a document commonly written in [HyperText](https://www.computerhope.com/jargon/h/hypertex.htm) Markup Language ([HTML](https://www.computerhope.com/jargon/h/html.htm)) that is accessible through the Internet or other network using an Internet [browser](https://www.computerhope.com/jargon/b/browser.htm). A web page is accessed by entering a URL address and may contain text, graphics, and [hyperlinks](https://www.computerhope.com/jargon/h/hyperlin.htm) to other web pages and files.

# CLIENT SERVER ARCHITECTURE

**Client-server architecture**, architecture of a [computer](https://www.britannica.com/technology/computer) [network](https://www.britannica.com/technology/computer-network) in which many [clients](https://www.britannica.com/technology/client) (remote processors) request and receive service from a centralized [server](https://www.britannica.com/technology/server) (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the [hardware](https://www.britannica.com/technology/hardware-computing) and [software](https://www.britannica.com/technology/software)) that is providing the service. Clients are often situated at [workstations](https://www.britannica.com/technology/workstation) or on [personal computers](https://www.britannica.com/technology/personal-computer), while servers are located elsewhere on the network, usually on more powerful machines. This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform. In [hospital](https://www.britannica.com/science/hospital) [data processing](https://www.britannica.com/technology/data-processing), for example, a client computer can be running an application program for entering patient information while the server computer is running another program that manages the [database](https://www.britannica.com/technology/database) in which the information is permanently stored. Many clients can access the server’s information simultaneously, and, at the same time, a client computer can perform other tasks, such as sending [e-mail](https://www.britannica.com/technology/e-mail). Because both client and server computers are considered intelligent devices, the client-server model is completely different from the old “[mainframe](https://www.britannica.com/technology/mainframe)” model, in which a centralized mainframe computer performed all the tasks for its associated “dumb” terminals.

**CHAPTER 2**

**HTML**



# INTRODUCTION TO HTML

HTML is the standard markup language for creating Web pages.

* + - HTML stands for Hyper Text Markup Language
    - HTML describes the structure of Web pages using markup
    - HTML elements are the building blocks of HTML pages
    - HTML elements are represented by tags
    - HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
    - Browsers do not display the HTML tags, but use them to render the content of the page

# HISTORY OF HTML

The history of hypertext markup language is a strange and interesting tale. From its simple start as an online subset of SGML through political maneuverings of the huge browser companies to its current piecemeal – but growing – compatibility, the language has weathered a storm of growth, abuse, and innovation. The idea behind HTML was a modest one. When Tim Berners- Lee was putting together his first elementary browsing and authoring system for the Web, he created a quick little hypertext language that would serve his purposes. He imagined dozens, or even hundreds, of hypertext formats in the future, and smart clients that could easily negotiate and translate documents from servers across the Net.

In 1993, a debate was exploding on the fledgling HTML mailing list, and finally a college student named Marc Andreessen added <img> to his Mosaic browser. People objected, saying it was too limited. They wanted <include>or <embed>, which would allow you to add any sort of media to a Web page with the much-touted content negotiation used on the client. That was too big a project, according to Marc, and he needed to ship ASAP. Mosaic went with <img>, and it would be years before including media in a page using <embed>, or <applet>, or <object> would come to the surface again.

Something needed to give. If things kept up the way they were going, Netscape and Microsoft would eventually have two completely proprietary versions of HTML, but with no way of supporting the utopian vision of content negotiation.

# HTML TAGS

HTML tags are element names surrounded by angle brackets:

<tagname>content goes here...</tagname>

* + - HTML tags normally come **in pairs** like <p> and </p>
    - The first tag in a pair is the **start tag,** the second tag is the **end tag**
    - The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

## Web Browsers

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.

HTML Page Structure

Below is a visualization of an HTML page structure:

<html>

<head>

<title>Page title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph</p>

<p>This is another paragraph</p>

</body>

</html>

## Example.2.3.1

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

## Output:

**My First Heading** My first paragraph.

**Example Explained**

* The <html> element is the root element of an HTML page
* The <head> element contains meta information about the document
* The <title> element specifies a title for the document
* The <body> element contains the visible page content
* The <h1> element defines a large heading
* The <p> element defines a paragraph

## HTML Editors

Write HTML Using Notepad or TextEdit

Web pages can be created and modified by using professional HTML editors.

However, for learning HTML we recommend a simple text editor like Notepad (PC) or TextEdit (Mac).

We believe using a simple text editor is a good way to learn HTML.

Follow the four steps below to create your first web page with Notepad or TextEdit. Step 1: Open Notepad (PC)

## Windows 8 or later:

Open the **Start Screen** (the window symbol at the bottom left on your screen). Type **Notepad**.

## Windows 7 or earlier:

Open **Start** > **Programs > Accessories > Notepad**

Step 2: Write Some HTML

Write or copy some HTML into Notepad.

<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

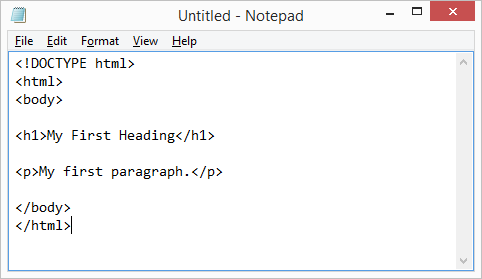


Figure 2.1 HTML Code

Step 3: Save the HTML Page

Save the file on your computer. Select **File > Save as** in the Notepad menu.

Name the file **"index.htm"** and set the encoding to **UTF-8** (which is the preferred encoding for HTML files).

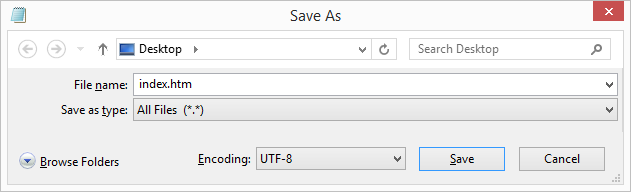


Figure 2.2 Saving HTML Code Step 4: View the HTML Page in Your Browser

Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").

The result will look much like this:

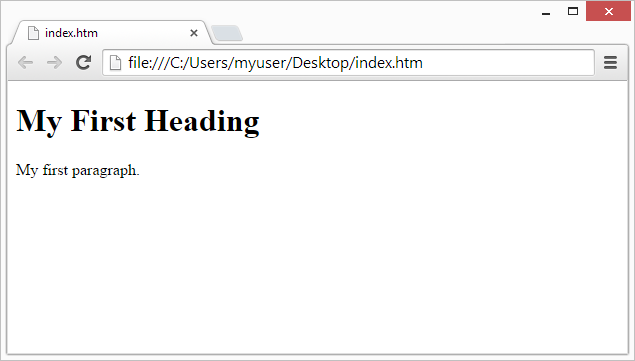


Figure 2.3 Output of HTML Code

## HTML Documents

The HTML document itself begins with <html> and ends with </html>.

The visible part of the HTML document is between <body> and </body>.

## Example:2.3.2

<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

## Output:

**My First Heading**

My first paragraph.

## HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

## <h1> defines the most important heading. <h6> defines the least important

<h1>This is heading 1</h1>

Example:2.3.3

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

## Output:

**This is heading 1 This is heading 2 This is heading 3**

## HTML Paragraphs

HTML paragraphs are defined with the <p> tag:

Example:2.3.4

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

## HTML Links

## 

## HTML links are defined with the <a> tag:

Example:2.3.5

## <a href="https:[//w](http://www.w3schools.com/)ww[.w3schools.com">This](http://www.w3schools.com/) is a link</a>

* + 1. **HTML Images**

HTML images are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

Example: 2.3.6

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

## HTML Buttons

HTML buttons are defined with the <button> tag: **<button>Click me</button>**

Example: 2.3.7

* + 1. **HTML Lists**

HTML lists are defined with the <ul> (unordered/bullet list) or the <ol> (ordered/numbered list) tag, followed by <li> tags (list items):

Example:2.3.8

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

## HTML Elements

An HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between:

<tagname>Content goes here...</tagname>

The HTML **element** is everything from the start tag to the end tag:

<p>My first paragraph.</p>

**2.4 HTML TABLES**

Defining an HTML Table

An HTML table is defined with the <table> tag.

Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

Example:2.4.1

<html>

<body>

<h2>Basic HTML Table</h2>

<table style="width:100%">

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<tr>

<td>Anu</td>

<td>Smith</td>

<td>23</td>

</tr>

<tr>

<td>John</td>

<td>Nick</td>

<td>34</td>

</tr>

<tr>

<td>John</td>

<td>Dov</td>

<td>40</td>

</tr>

</table>

</body>

</html>

Table 2.1 Basic HTML

|  |  |  |
| --- | --- | --- |
| **Firstname** | **Last Name** | **Age** |
| Anu | Smith | 23 |
| John | Nick | 34 |
| John | Dov | 40 |

## HTML Table - Adding a Border

If you do not specify a border for the table, it will be displayed without borders.

A border is set using the CSS border property:

table, th, td {

border: 1px solid black;

}

## HTML Table - Collapsed Borders

If you want the borders to collapse into one border, add the CSS border-collapse property:

Example:2.4.2

## table, th, td {

**border: 1px solid black; border-collapse: collapse;**

## }

## 2.4.3 HTML Table - Adding Cell Padding

Cell padding specifies the space between the cell content and its borders.

If you do not specify a padding, the table cells will be displayed without padding.

To set the padding, use the CSS padding property:

Example:2.4.3

th, td {

padding: 15px;

}

## HTML Table - Left-align Headings

By default, table headings are bold and centered.

To left-align the table headings, use the CSS text-align property:

Example:2.4.4

th {

text-align: left;

}

## HTML Table - Adding Border Spacing

Border spacing specifies the space between the cells.

To set the border spacing for a table, use the CSS border-spacing property:

Example:2.4.5

table {a

border-spacing:

}

# 2.5HTML FORMS

The <form> Element

The HTML <form> element defines a form that is used to collect user input:

<form>

.

*form elements*

*.*

</form>

The <input> Element

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the **type** attribute. Here are some examples

Table 2.2 HTML Forms

|  |  |
| --- | --- |
| **Type** | **Description** |
| <input type="text"> | Defines a one-line text input field |
| <input type="radio"> | Defines a radio button (for selecting one of many choices) |
| <input type="submit"> | Defines a submit button (for submitting the form) |

## Text Input

<input type="text"> defines a one-line input field for **text input**:

Example:2.5.1

<form>

First name:<br>

<input type="text" name="firstname"><br> Last name:<br>

<input type="text" name="lastname">

</form> **Text Input** First name:

Last name:

## Radio Button Input

## input type="radio"> defines a radio button.

Radio buttons let a user select ONE of a limited number of choices:

## Example:2.5.2

<form>

<input type="radio" name="gender" value="male" checked> Male<br>

<input type="radio" name="gender" value="female"> Female<br>

<input type="radio" name="gender" value="other"> Other

</form>

This is how the HTML code above will be displayed in a browser:

Male Female Other



## The Submit Button

<input type="submit"> defines a button for **submitting** the form data to a **form-handler**. The form-handler is specified in the form's **action** attribute:

Example:2.5.3

<form action="/action\_page.php">

First name:<br>

<input type="text" name="firstname" value="Mickey"><br> Last name:<br>

<input type="text" name="lastname" value="Mouse"><br><br>

<input type="submit" value="Submit">

</form>

**HTML Forms**

First name:

Last name:

Submit

**CHAPTER 3**

**CSS**



# INTRODUCTION

* + - CSS stands for Cascading Style Sheets
    - CSS describes how HTML elements are to be displayed on screen, paper, or in other media
    - CSS saves a lot of work. It can control the layout of multiple web pages all at once
    - External stylesheets are stored in CSS files

# HISTORY OF CSS

Before Cascading Style Sheets (CSS) there was very little that could be done to change the design of a web page. While Hyper Text Markup Language (HTML) creates documents for the World Wide Web, it was specifically designed to hold the content of a web page. Housed in a separate file, CSS adds the style and design to a web page. The term cascading comes from the ability to combine multiple CSS files to determine the style for one page.

As more people started using HTML, the demand grew for more design capabilities, which would allow developers to control how web documents looked. But browsers offered limited capabilities for styling. In 1993 NCSA Mosaic was released, making the web more popular than ever, but it only offered limited capability to change fonts and colors.

In October 1994, Tim Berners-Lee formed the World Wide Web Consortium, (W3C) at the Massachusetts Institute of Technology Laboratory for Computer Science. The W3C has members that are government entities, businesses, educational institutions and individuals.

Although it took 3 years for any browser to come close to fully implementing CSS, August 1996 Microsoft Internet Explorer became the first browser to support CSS. Netscape followed suit in supporting CSS, but also implemented an alternative Javascript Style Sheets, which were never fully completed, and are now deprecated. To this day, there are differences in the way CSS is implemented in different browsers, leading developers to use hacks to make web pages look consistent in different browsers.

## Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes**.**

# CSS SYNTAX

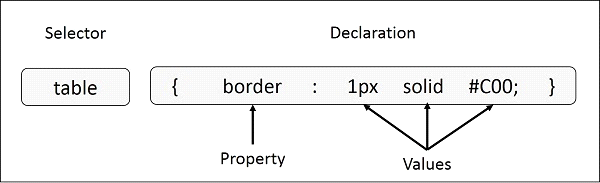


Fig 3.1CSS SYNTAX

**Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

**Property** − A property is a type of attribute of an HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

**Value** − Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

**Example:3.1** − You can define a table border as follows − table{ border :1px solid #C00; }

# TYPES OF CSS

There are three types of css-

* + - Internal style sheet
    - External style sheet
    - Inline style

## Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML

page:

Example:3.2

<head>

<style> body {

background-color: linen;

}

h1 {

color: maroon; margin-left: 40px;

}

</style>

</head>

## External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Here is how the "mystyle.css" looks: body {

background-color: lightblue;

}

h1 {

color: navy; margin-left: 20px;

}

## Inline Styles

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>

# CSS COLORS

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

## CSS colors-hex codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

Color Color HEX black #000000

red #FF0000 green #00FF00 blue #0000FF yellow #FFFF00 white #FFFFFF

## CSS Background

The CSS background properties are used to define the background effects for elements. CSS background properties:

* + - * background-color
      * background-image
      * background-repeat
      * background-attachment
      * background-position

## EXAMPLE:3.4

body {

background-color: lightblue;

}

## CSS Borders

css border properties:

The CSS border properties allow you to specify the style, width, and color of an element's border.

The border-style property specifies what kind of border to display. The following values are allowed:

* + - * dotted - Defines a dotted border
      * dashed - Defines a dashed border
      * solid - Defines a solid border
      * double - Defines a double border
      * groove - Defines a 3D grooved border. The effect depends on the border-color value
      * ridge - Defines a 3D ridged border. The effect depends on the border-color value
      * inset - Defines a 3D inset border. The effect depends on the border-color value
      * outset - Defines a 3D outset border. The effect depends on the border-color value
      * none - Defines no border
      * hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

## CSS TEXT

* + - * The color property is used to set the color of a text.
      * The direction property is used to set the text direction.
      * The text-align property is used to align the text of a document.

The following example demonstrates how to set the text color. Possible value could be any color name in any valid format.

Example:3.5

<html>

<head>

</head>

<body>

<p style = "color:red;">

This text will be written in red.

</p>

</body>

</html>

## Font Family

The font family of a text is set with the font-family property.

## EXAMPLE:3.6

p {

font-family: "Times New Roman", Times, serif;

} sss

## CSS Tables TABLE BORDER

To specify table borders in CSS, use the border property**. SYNTAX**

table, th, td {

border: 1px solid black;

}

**CHAPTER 4**

**INTRODUCTION TO PYTHON**

Figure 4.1 PYTHON

# 4.1 INTRODUCTION

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is a dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). It was created by Guido van Rossum and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions: Python 2 and Python 3. Both are quite different.

## 4.2 HISTORY OF PYTHON

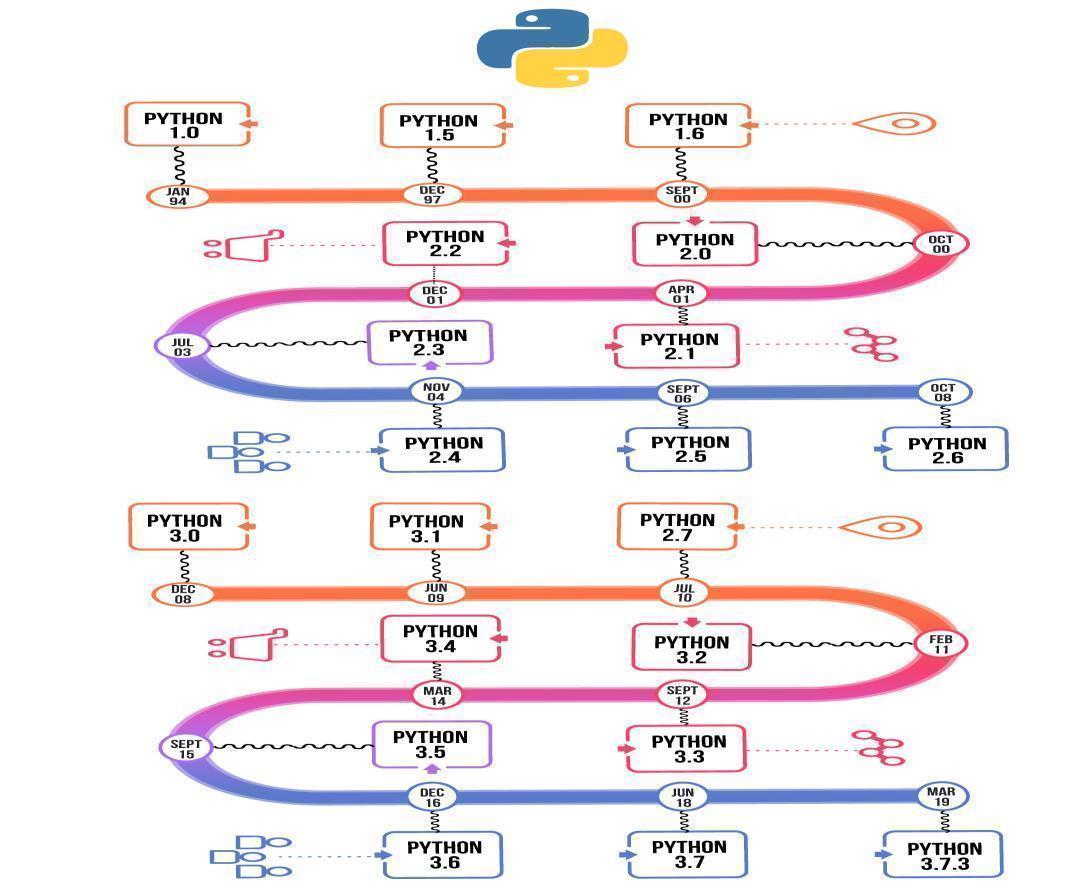
In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in the Netherlands. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language in which Python is said to have succeeded is ABC Programming Language, which had interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did was really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language that had removed all the flaws. The inspiration for the name came from BBC’s TV Show – ‘Monty Python’s Flying Circus’, as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the “Benevolent dictator for life” (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox. The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data types exception handling and functions. 16 Following are the illustrations of different versions of Python along with the timeline.

Figure 4.2 Python History

FIGURE 4.2

FIGURE 4.2

**4.2.1 WHY PYTHON**

Python is consistently rated as one of the world's most popular programming languages. Python is fairly easy to learn, so if you are starting to learn any programming language then Python could be your great choice. Today various Schools, Colleges and Universities are teaching Python as their primary programming language. There are many other good reasons which makes Python as the top choice of any programmer:

* Python is Open Source which means it's available free of cost.
* Python is simple and so easy to learn.
* Python is versatile and can be used to create many different things.
* Python has powerful development libraries including AI, ML etc.
* Python is much in demand and ensures high salary

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

**Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented** − Python supports the ObjectOriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## 4.2.2 APPLICATION

**1. Web Development**

Python can be used to make web-applications at a rapid rate. Why is that? It is because of the frameworks Python uses to create these applications. There is common-backend logic that goes into making these frameworks and a number of libraries that can help integrate 18 protocols such as HTTPS, FTP, SSL etc. and even help in the processing of JSON, XML, E-Mail and so much more.

Figure 4.3 Web development

Some of the most well-known frameworks are Django, Flask, Pyramid.Why Use A Framework?The security, scalability, convenience that they provide is commendable if we compare it to starting the development of a website from scratch.

**2. Game Development**

Python is also used in the development of interactive games. There are libraries such as PySoy which is a 3D game engine supporting Python 3, PyGame which provides functionality and a library for game development. Games such as Civilization-IV, Disney’s Toontown Online, Vega Strike etc. have been built using Python.

Figure 4.4 Game Development

**3. Machine Learning and Artificial Intelligence**

Machine Learning and Artificial Intelligence are the talks of the town as they yield the most promising careers for the future. We make the computer learn based on past experiences through the data stored or better yet, create algorithms which makes the computer learn by itself. The programming language that mostly everyone chooses? It’s Python. Why? Support for these domains with the libraries that exist already such as Pandas, ScikitLearn, NumPy and so many more.

Figure 4.5 AI & ML

Learn the algorithm, use the library and you have your solution to the problem. It is that simple. But if you want to go the hardcore way, you can design your own code which yields a better solution, which still is much easier when we compare it to other languages.

**4. Data Science and Data Visualization**

Data is money if you know how to extract relevant information which can help you take calculated risks and increase profits. You study the data you have, perform operations and extract the information required. Libraries such as Pandas, NumPy help you in extracting information.

Figure 4.6 Data Analysis 

You can even visualize the data libraries such as Matplotlib, Seaborn, which are helpful in plotting graphs and much more. This is what Python offers you to become a Data Scientist.

**5. Desktop GUI**

We use Python to program desktop applications. It provides the Tkinter library that can be used to develop user interfaces. There are some other useful toolkits such as the wxWidgets, Kivy, PYQT that can be used to create applications on several platforms.

Figure 4.7 GUI

You can start out with creating simple applications such as Calculators, To-Do apps and go ahead and create much more complicated applications.

**6. Web Scraping Applications**

Python is a savior when it comes to pulling a large amount of data from websites which can then be helpful in various real-world processes such as price comparison, job listings, research and development and much more.

**7. Audio and Video Applications**

We use Python to develop applications that can multi-task and also output media. Video and audio applications such as Tim Player, C play have been developed using Python libraries. They provide better stability and performance in comparison to other media players.

Figure 4.8 Media

## 4.2.3 FEATURES

There are many features in Python, some of which are discussed below as follows:

**1. Free and Open Source**

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

**2. Easy to code**

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

**3. Easy to Read**

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

**4. Object-Oriented Language**

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc. 5. GUI Programming Support Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

**6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

**7. Extensible feature**

Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

**8. Easy to Debug**

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to interpret Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

**9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

**10. Python is an Integrated language**

Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.

**11. Interpreted Language**

Python is an Interpreted Language because Python code is executed line by line at a time. Unlike other languages C, C++, Java, etc. there is no need to compile Python code; this makes it easier to debug our code. The source code of Python is converted into an immediate form called bytecode.

**12. Large Standard Library**

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

**13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

**14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags

**CHAPTER 5**

**BASICS OF PYTHON**



# 5.1 LISTS

# Python Lists are just like dynamically sized arrays, declared in other languages (vector in C++ and Array List in Java). In simple language, a list is a collection of things, enclosed in [ ] and separated by commas. The list is a sequence data type which is used to store the collection of data

# Example of list in Python

# Here we are creating a Python List using [].

# Var = ["hello , "hii"]

# print(Var)

# #Output: ["hello", "hii"]

# Lists are the simplest containers that are an integral part of the Python language. Lists need not be homogeneous always which makes it the most powerful tool in Python. A single list may contain Data Types like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.

Table 5.1 List Method

|  |  |
| --- | --- |
| Function | Description |
| Append() | Add an element to the end of the list |
| Extend() | Add all elements of a list to another list |
| Insert() | Insert an item at the defined index |
| Remove() | Removes an item from the list |
| Clear() | Removes all items from the list |
| Index() | Returns the index of the first matched item |
| Count() | Returns the count of the number of items passed as 26 an argument |
| Sort() | Sort items in a list in ascending order |
| Reverse() | Reverse the order of items in the list |
| copy() | Returns a copy of the list |

# Table 5.2 Built-in functions with List

|  |  |
| --- | --- |
| Function | Description |
| reduce() | apply a particular function passed in its argument to all of the list elements stores the intermediate result and only returns the final summation value |
| sum() | Sums up the numbers in the list |
| ord() | Returns an integer representing the Unicode code point of the given Unicode character |
| cmp() | This function returns 1 if the first list is “greater” than the second list |
| max() | return maximum element of a given list min() return minimum element of a given list |
| all() | Returns true if all element is true or if the list is empty |
| any() | return true if any element of the list is true. if the list is empty, return false |
| len() | Returns length of the list or size of the list |
| enumerate() | Returns enumerate object of the list |
| accumulate() | apply a particular function passed in its argument to all of the list elements returns a list containing the intermediate results |
| filter() | tests if each element of a list is true or not |
| map() | returns a list of the results after applying the given function to each item of a given iterable. |
| lambda() | This function can have any number of arguments 27 but only one expression, which is evaluated and returned. |

# 

# SLICING

# We can get substrings and sub lists using a slice. In Python List, there are multiple ways to print the whole list with all the elements, but to print a specific range of elements from the list, we use the Slice operation.

# Slice operation is performed on Lists with the use of a colon(:),

# To print elements from beginning to a range use:

# [: Index]

# To print elements from end-use:

# [:-Index]

# To print elements from a specific Index till the end use

# [Index:] To print the whole list in reverse order, use

# [::-1]

# Note – To print elements of List from rear-end, use Negative Indexes.

# Figure 5.1 Slicing in Python

# 5.2 TUPLE

# Python Tuple is a collection of objects separated by commas. In some ways, a tuple is similar to a list in terms of indexing, nested objects, and repetition but a tuple is immutable, unlike lists which are mutable.

# Creating Python Tuples

# To create a tuple we will use () operators.

# var = ("Hello", "Hii")

# print(var)

# Output:

# ('Hello', 'Hii’)

# Note: In case you're generating a tuple with a single element, make sure to add a comma after the element.

# Figure 5.2 Tuples in Python

# 5.3 SETS

# A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

# Figure 5.3 Set in Python

# Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

# 5.4 DICTIONARY

# Python Dictionary is used to store the data in a key-value pair format. The dictionary is the data type in Python, which can simulate the real-life data arrangement where some specific value 30 exists for some particular key. It is a mutable data-structure. The dictionary is defined into element Keys and values.

# Keys must be a single element

# Value can be any type such as list, tuple, integer, etc.

# In other words, we can say that a dictionary is the collection of key value pairs where the value can be any Python object. In contrast, the keys are the immutable Python object, i.e., Numbers, string, or tuple. Creating the dictionary the dictionary can be created by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:). The syntax to define the dictionary is given below.

# Syntax:

# 1. Dict = {"Name": "Tom", "Age": 22}

# In the above dictionary Dict, The keys Name and Age are the string that is an immutable object.

# 

# 

# 

# 

# 

# 

# 

# 

# Figure 5.4 Dictionary in Python

# 5.5 FUNCTIONS

# Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

# Figure 5.5 Functions in Python

# Syntax: Python Functions

# METHOD 1 FUNCTION WITHOUT ARGUMENT AND WITH NO RETURN VALUE

# def sum ():

# a=int(input(“enter a”))

# b=int(input(“enter b”))

# c=a+b

# print(c)

# sum ()

# METHOD 2 FUNCTION WITH ARGUMENT AND WITH NO RETURN VALUE

# def sum(a,b):

# c=a+b

# print(c)

# x=int(input(“enter no”))

# y=int(input(“enter no”))

# sum(x,y)

# METHOD 3 FUNCTION WITHOUT ARGUMENT AND WITH RETURN VALUE

# def sum():

# a=int(input(“enter a”))

# b=int(input(“enter b”))

# c=a+b

# return c

# z=sum()

# sum(z)

# METHOD 4 FUNCTION WITH ARGUMENT AND WITH RETURN VALUE

# def sum(a,b):

# c=a+b

# return c

# z=sum(4,3)

# print(z)

# 5.6 INHERITANCE

# Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch. 33 In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In this section of the tutorial, we will discuss inheritance in detail. In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name.

# TYPES OF INHERITANCE

# 1. Single Inheritance:

# Single inheritance enables a derived class to inherit properties from a single parent class, thus enabling code re-usability and the addition of new features to existing code.

# Figure 5.6 Single Inheritance

# 2. Multiple Inheritance:

# When a class can be derived from more than one base class this type of inheritance is called multiple inheritances. In multiple inheritances, all the features of the base classes are inherited into the derived class.

# Figure 5.7 Multiple Inheritance

# 3. Multilevel Inheritance :

# In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class. This is similar to a relationship representing a child and a grandfather.

# Figure 5.8 Multilevel Inheritance

# 4. Hierarchical Inheritance:

# When more than one derived class is created from a single base this type of inheritance is called hierarchical inheritance. In this program, we have a parent (base) class and two child (derived) classes.

# Figure 5.9 Hierarchical Inheritance

# 5. Hybrid Inheritance:

# Inheritance consisting of multiple types of inheritance is called hybrid inheritance.

# 

# Figure 5.10 Hybrid Inheritance

**CHAPTER 6**

**SQL**



# 6.1 INTRODUCTION TO SQL

# Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS

## 6.1.1 DDL: Data Definition Language

## This includes changes to the structure of the table like creation of table, altering table, deleting a table etc. All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Table 6.1 Data Definition Language

|  |  |
| --- | --- |
| **Command** | **Description** |
| Create | to create new table or database |
| Alter | for alteration |
| Truncate | delete data from table |
| Drop | to drop a table |
| Rename | to rename a table |

## 6.1.2 DML: Data Manipulation Language

## DML commands are used for manipulating the data stored in the table and not the table itself.

DML commands are not auto-committed. It means changes are not permanent to the database, they can be rolled back.

Table 6.2 Data Manipulation Language

|  |  |
| --- | --- |
| **Command** | **Description** |
| Insert | to insert a new row |
| Update | to update existing row |
| Delete | to delete a row |
| Merge | merging two rows or two tables |

## 

## 6.1.3 TCL: Transaction Control Language

These commands are to keep a check on other commands and their effect on the database. These commands can annul changes made by other commands by rolling the data back to its original state. It can also make any temporary change permanent.

Table 6.3Transaction Control Language

|  |  |
| --- | --- |
| **Command** | **Description** |
| Commit | to permanently save |
| Rollback | to undo change |
| Savepoint | to save temporarily |

## 6.1.4 DCL: Data Control Language

Data control languages are the commands to grant and take back authority from any database user.

Table 6.4Data Control Language

|  |  |
| --- | --- |
| **Command** | **Description** |
| Grant | grant permission of right |
| Revoke | take back permission. |
| Select | retrieve records from one or more table |

# 6.2 SQL LANGUAGE ELEMENTS

These topics provide detailed descriptions of the language elements and conventions of Sybase IQ SQL.

## [Keywords](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452827349.html)

## Each SQL statement contains one or more keywords.

## Identifiers

## Identifiers are names of objects in the database, such as user IDs, tables, and columns

## [Strings](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452830286.html)

Strings are either literal strings, or expressions with CHAR or VARCHAR data types.

## [Expressions](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452832083.html)

Expressions are formed from several different kinds of elements, such as constants, column names, SQL operators, and subqueries.

## [SearchConditions](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452851990.html)

Conditions are used to choose a subset of the rows from a table, or in a control statement such as an IF statement to determine control of flow.

## [SpecialValues](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452878302.html)

Special values can be used in expressions, and as column defaults when creating tables.

## [Variables](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452889396.html)

Sybase IQ supports local variables, connection-level variables, and global variables.

## [Comments](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452895506.html)

Use comments to attach explanatory text to SQL statements or statement blocks. The database server does not execute comments.

## [NULLValue](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452896896.html)

NULL to specify a value that is unknown, missing, or not applicable.

## [Strings](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452830286.html)

Strings are either literal strings, or expressions with CHAR or VARCHAR data types.

## [Expressions](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452832083.html)

Expressions are formed from several different kinds of elements, such as constants, column names, SQL operators, and subqueries.

## [Search Conditions](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452851990.html)

Conditions are used to choose a subset of the rows from a table, or in a control statement such as an IF statement to determine control of flow.

## [Special Values](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452878302.html)

Special values can be used in expressions, and as column defaults when creating tables.

[**Variables**](http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc38151.1540/doc/html/san1278452889396.html)**SQL** is incredibly powerful, and like every **well-made development tool**, it has a few commands which it’s vital for a good developer to know. Here are a few of the most important ones – each of these queries is consequential to almost every system that interacts with an [SQL database](https://bytescout.com/blog/introduction-into-sql-databases.html).

## 

## 6.2.1 Query for Retrieving Tables

This query can be run to retrieve the list of tables present in a database where the database is “My\_Schema”.

|  |  |
| --- | --- |
| 1 | SELECT \* FROM My\_Schema.Tables; |

## 6.2.2 Query for Selecting Columns from a Table

This is perhaps the most widely used SQL query. In the example below, we are extracting the “Student\_ID” column or attribute from the table “STUDENT”.

|  |  |
| --- | --- |
| 1 | SELECT Student\_ID FROM STUDENT; |

If you want to display all attributes from a particular table, this is the right query to use:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM STUDENT; |

## 6.2.3 Query for Outputting Data Using a Constraint

This query retrieves the specified attributes from the table on the constraint; Employee ID

=0000.

|  |  |
| --- | --- |
| 1 | SELECT EMP\_ID, NAME FROM EMPLOYEE\_TBL WHERE EMP\_ID = '0000'; |

## 6.2.4 Query for Outputting Sorted Data Using ‘Order By’

This query orders the results with respect to the attribute which is referenced to using “Order By” – so for example, if that attribute is an integer data type, then the result would either be sorted in ascending or descending order; likewise, if the datatype is a String then the result would be ordered in alphabetical order.

SELECT EMP\_ID, LAST\_NAME FROM EMPLOYEE WHERE CITY = 'Seattle'ORDER BY EMP\_ID;

The ordering of the result can also be set manually, using “asc ” for ascending and “desc” for descending.

SELECT EMP\_ID, LAST\_NAME FROM EMPLOYEE\_TBL WHERE CITY = 'INDIANAPOLIS'ORDER BY EMP\_ID asc;

## 6.2.5 Query for Outputting Sorted Data Using ‘Group By’

The ‘Group By’ property groups the resulting data according to the specified attribute.

|  |  |
| --- | --- |
| 1 | SELECT Name, Age FROM Patients WHERE Age > 40GROUP BY Age ORDER BY Name; |

## Queries for Data Manipulation Using Math Functions

There are a lot of built-in math functions like COUNT and AVG  which provide basic functionalities of counting the number of results and averaging them respectively.

## 6.2.6 Data Manipulation Using COUNT

This query displays the total number of customers by counting each customer ID. In addition, it groups the results according to the country of each customer.

|  |  |
| --- | --- |
| 1 | SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country; |

## 6.2.7 Data Manipulation Using SUM

SUM calculates the total of the attribute that is given to it as an argument.

|  |  |
| --- | --- |
| 1 | SELECT SUM(Salary)FROM Employee WHERE Emp\_Age< 30; |

## 6.2.8 Data Manipulation Using AVG

Simple – an average of a given attribute.

|  |  |
| --- | --- |
| 1 | SELECT AVG(Price)FROM Products; |

## 6.2.9 Query for Listing all Views

This query lists all the views available in the schema.

|  |  |
| --- | --- |
| 1 | SELECT \* FROM My\_Schema.views; |

## 2.2.10 Query for Creating a View

A view is a tailored table that is formed as a result of a query. It has tables and rows just like any other table. It’s usually a good idea to run queries as independent views, because this allows them to be retrieved later to view the query results, rather than computing the same command every time for a particular set of results.

|  |
| --- |
| CREATE VIEW Failing\_Students AS SELECT S\_NAME, Student\_ID FROM STUDENT WHERE GPA > 40; |

## 6.2.11 [Advantages of SQL](http://www.gcreddy.com/2014/07/sql-interview-questions-and-answers.html):

1. SQL Queries can be used to retrieve large amounts of records from a database quickly and efficiently.
2. SQL is used to view the data without storing the data into the object.
3. SQL joins two or more tables and show it as one object to user.
4. SQL databases use long-established standard, which is being adopted by ANSI &amp; ISO. Non-SQL databases do not adhere to any clear standard.
5. Using standard SQL it is easier to manage database systems without having to write substantial amount of code.
6. SQL restricts the access of a table so that nobody can insert the rows into the table.

## 6.2.12 Disadvantages of SQL

1. Interfacing an SQL database is more complex than adding a few lines of code.
2. When table is dropped view becomes inactive. It depends on the table objects.
3. Although SQL databases conform to ANSI &amp; ISO standards, some databases go for proprietary extensions to standard SQL to ensure vendor lock-in.
4. It is an object so it occupies space

**CHAPTER 7**

**DJANGO**

**7.1 INTRODUCTION** 

Django is a Python-based web framework which allows you to quickly create web applications without all of the installation or dependency problems that you normally will find with other frameworks. When you’re building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use.

**Why Django?**

1. It’s very easy to switch databases in the Django framework.

2. It has a built-in admin interface which makes it easy to work with it.

3. Django is a fully functional framework that requires nothing else.

4. It has thousands of additional packages available.

5. It is very scalable.

**Popularity of Django**

Django is used in many popular sites such as: Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic etc. There are more than 5k online sites based on the Django framework. ( Source ) Sites like Hot Frameworks assess the popularity of a framework by counting the number of GitHub projects and Stack Overflow questions for each platform, here Django is in 6th position. Web frameworks often refer to themselves as “opinionated” or “UN-opinionated” based on opinions about the right way to handle any particular task. Django is 54 somewhat opinionated, hence delivers in both worlds( opinionated & UN-opinionated ).

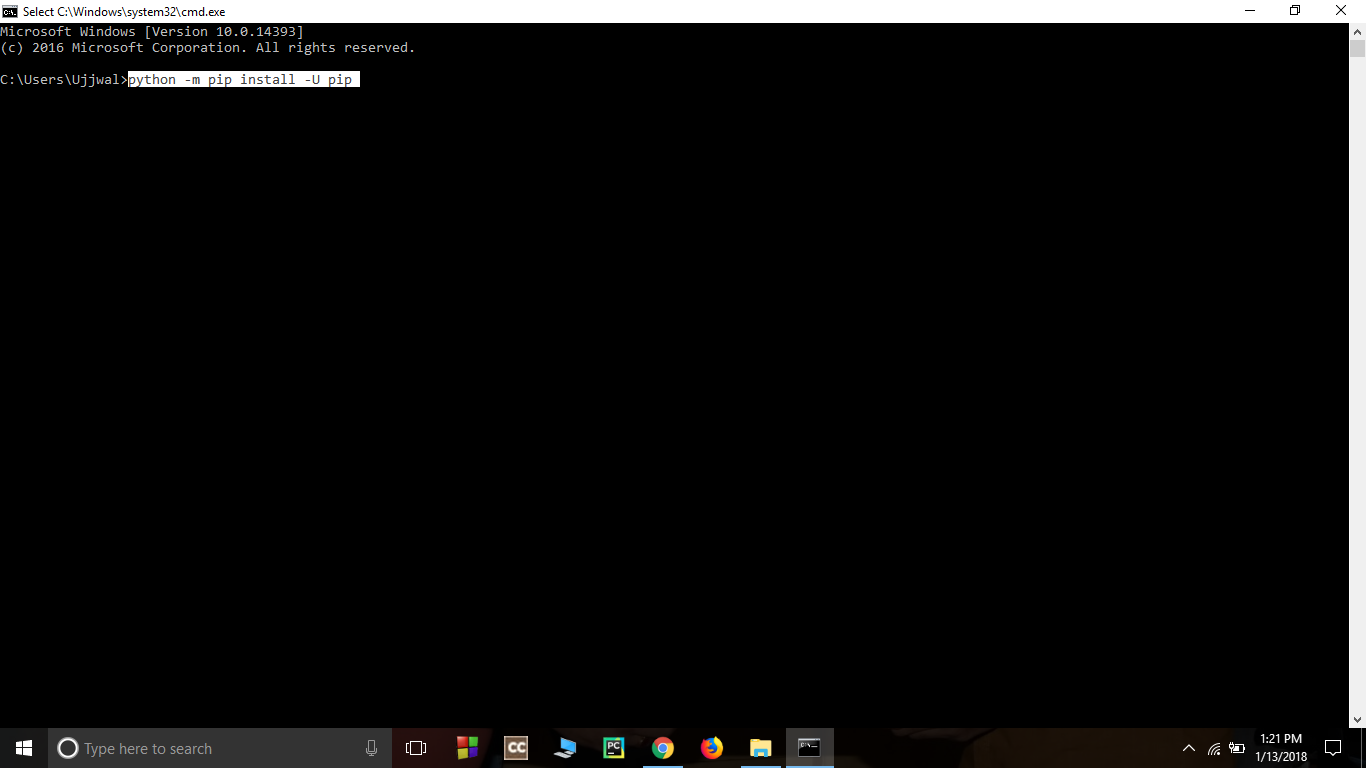
**7.2 INSTALLATION**

Install python3 if not installed in your system ( according to configuration of your system and OS).Try to download the latest version of python it’s python 3.11.0 this time.

**Note-** Installation of Django in Linux and Mac is similar, here I am showing it in windows for Linux and mac just open terminal in place of command prompt and go through the following commands.

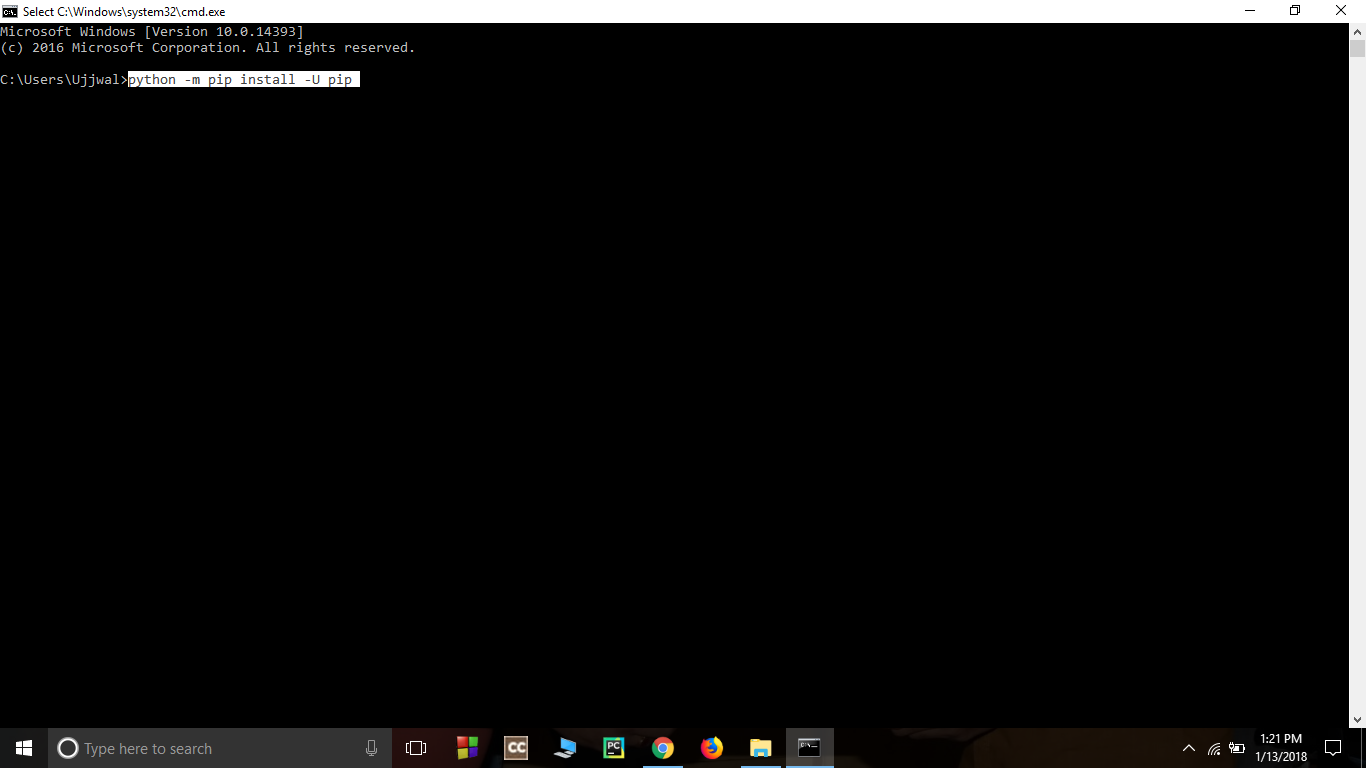
**Install pip-** Open command prompt and enter following command

python -m pip install -U pip

Figure 7.1 Install pip

**Install virtual environment-** Enter following command in cmd

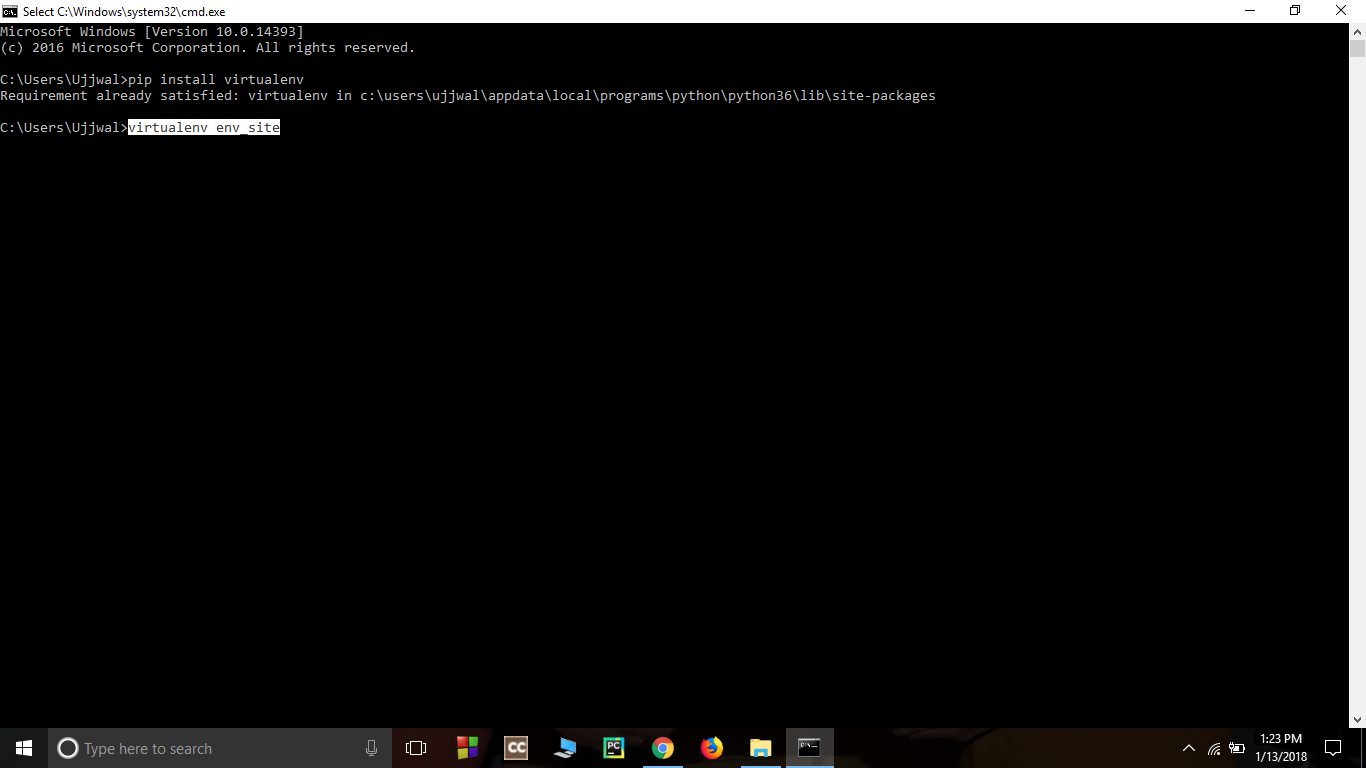
pip install virtualenv

Figure 7.2 Install virtual environment

**Set Virtual environment-** Setting up the virtual environment will allow you to edit the dependency which generally your system wouldn’t allow. Follow these steps to set up a virtual environment

1. Create a virtual environment by giving this command in cmd-

virtualenv env\_site

Figure 7.3 Set virtual environment

2. Change directory to env\_site by this command

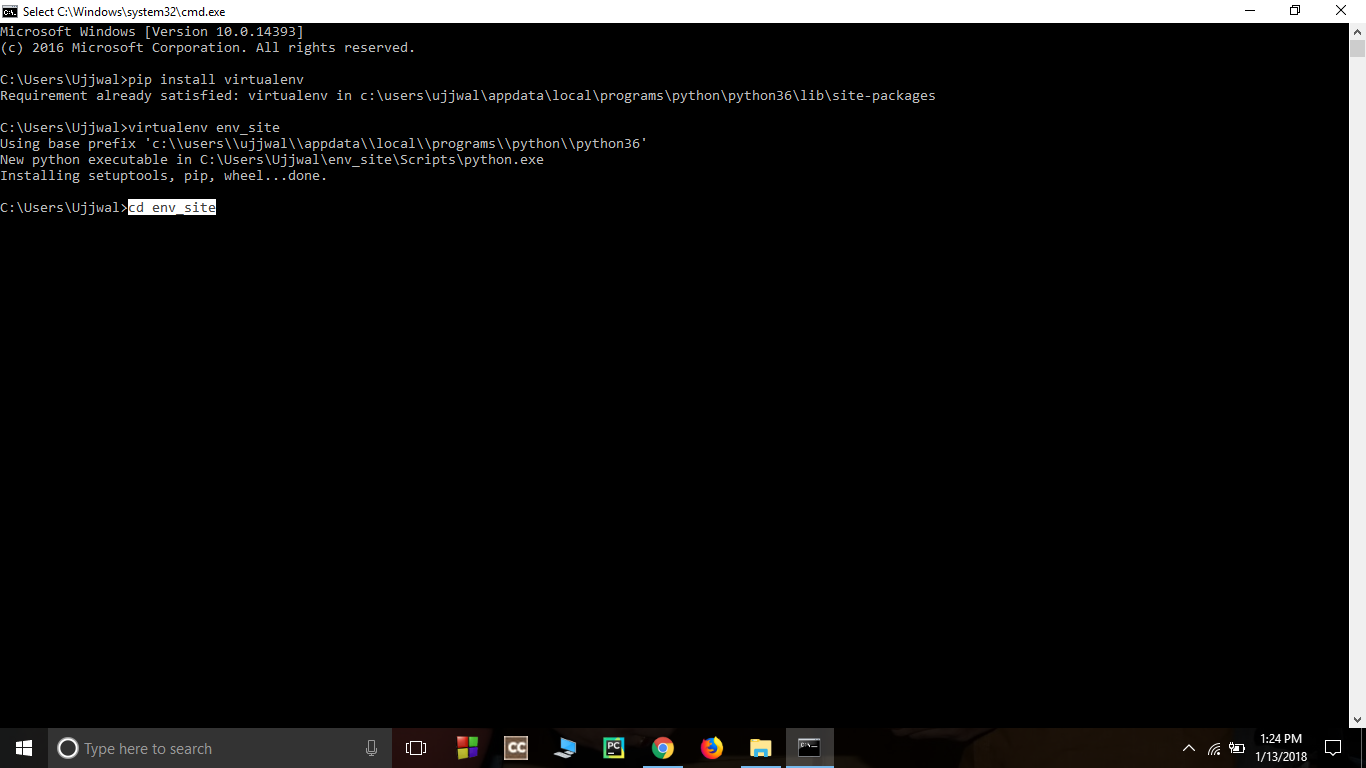
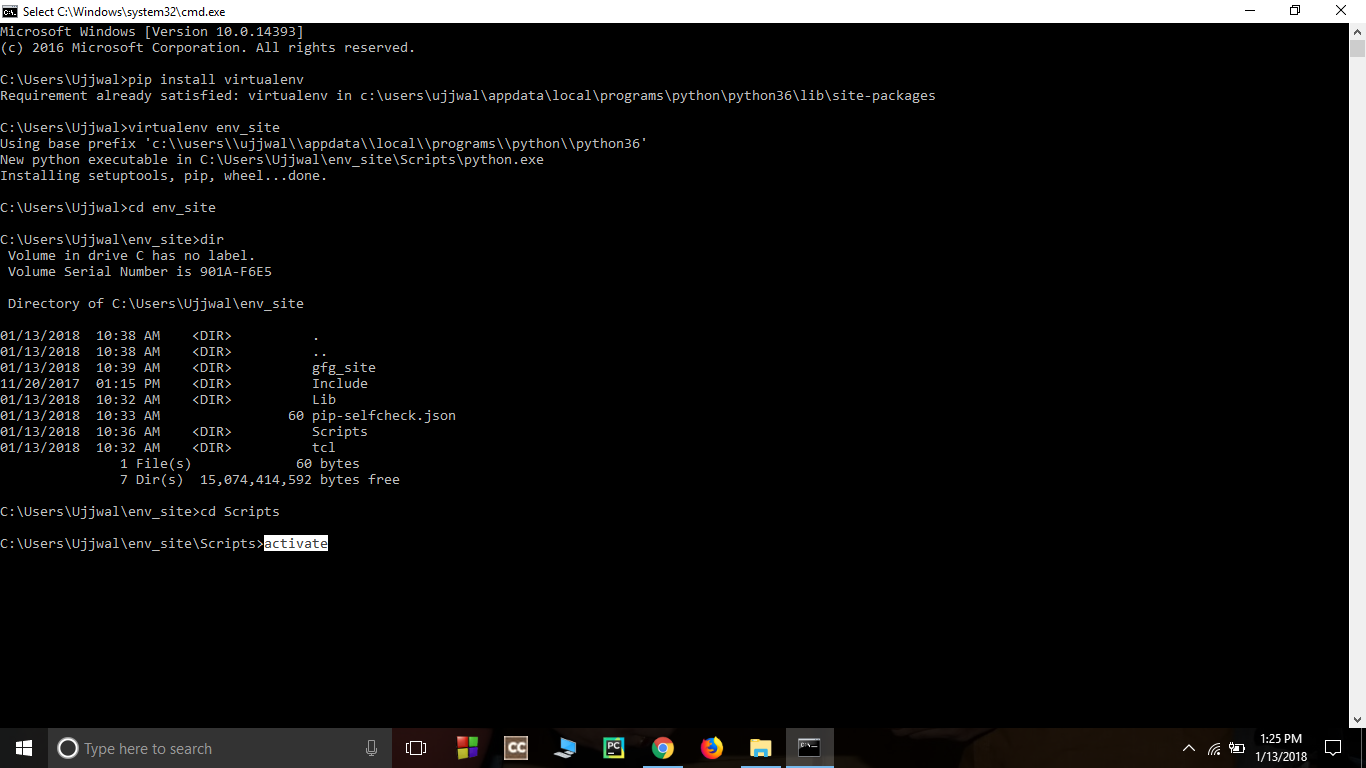
cd env\_site

Figure 7.4 Change Directory

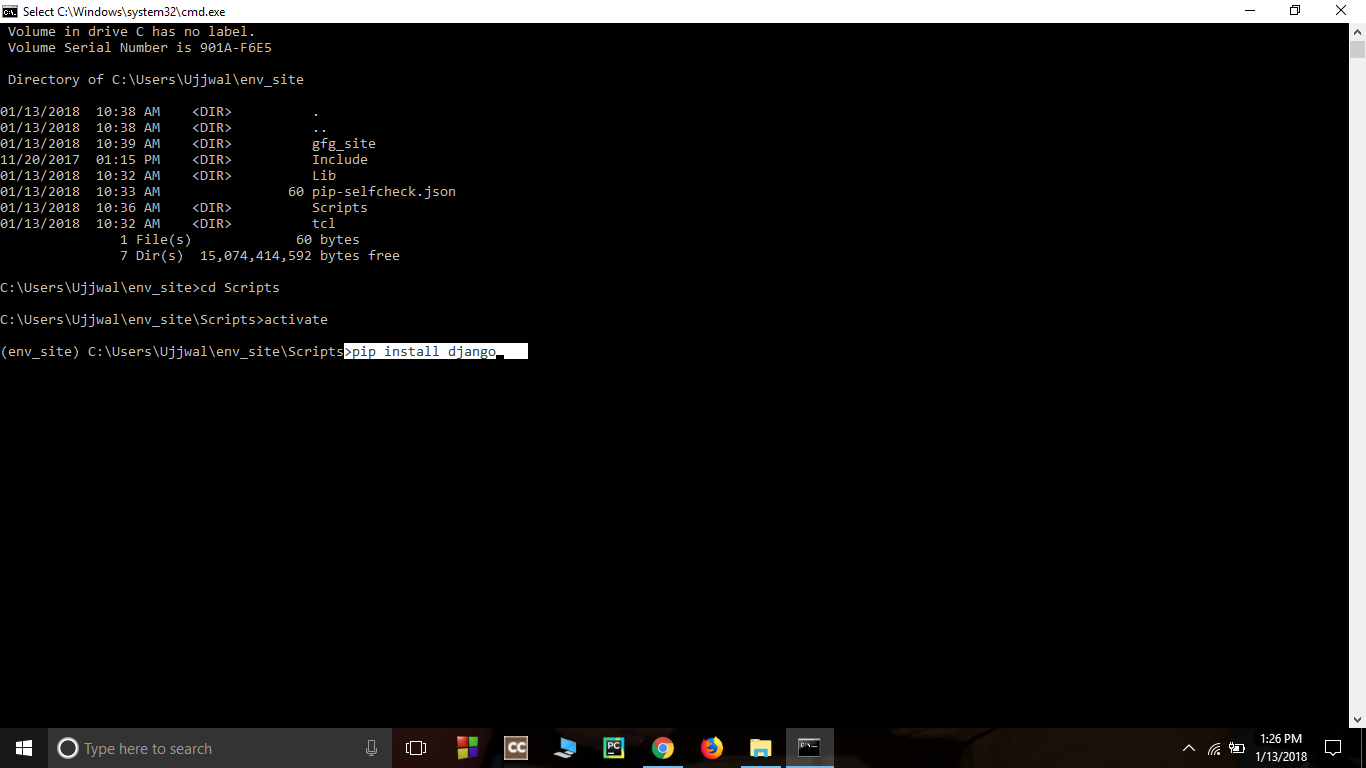
3. Go to Scripts directory inside env\_site and activate virtual environment-

cd Scripts activate

Figure 7.5 Activate Script

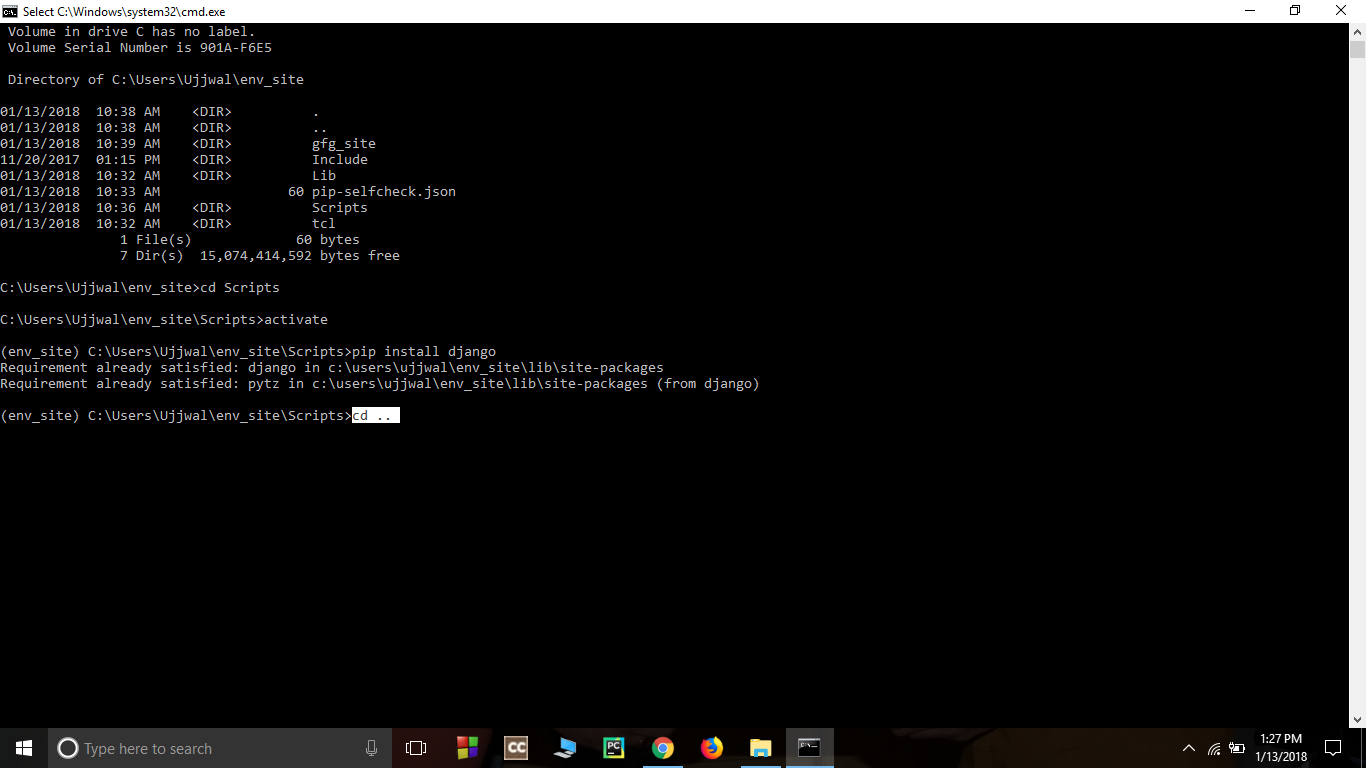
**Install Django -** Install django by giving following command

pip install django

Figure 7.6 Install Django 

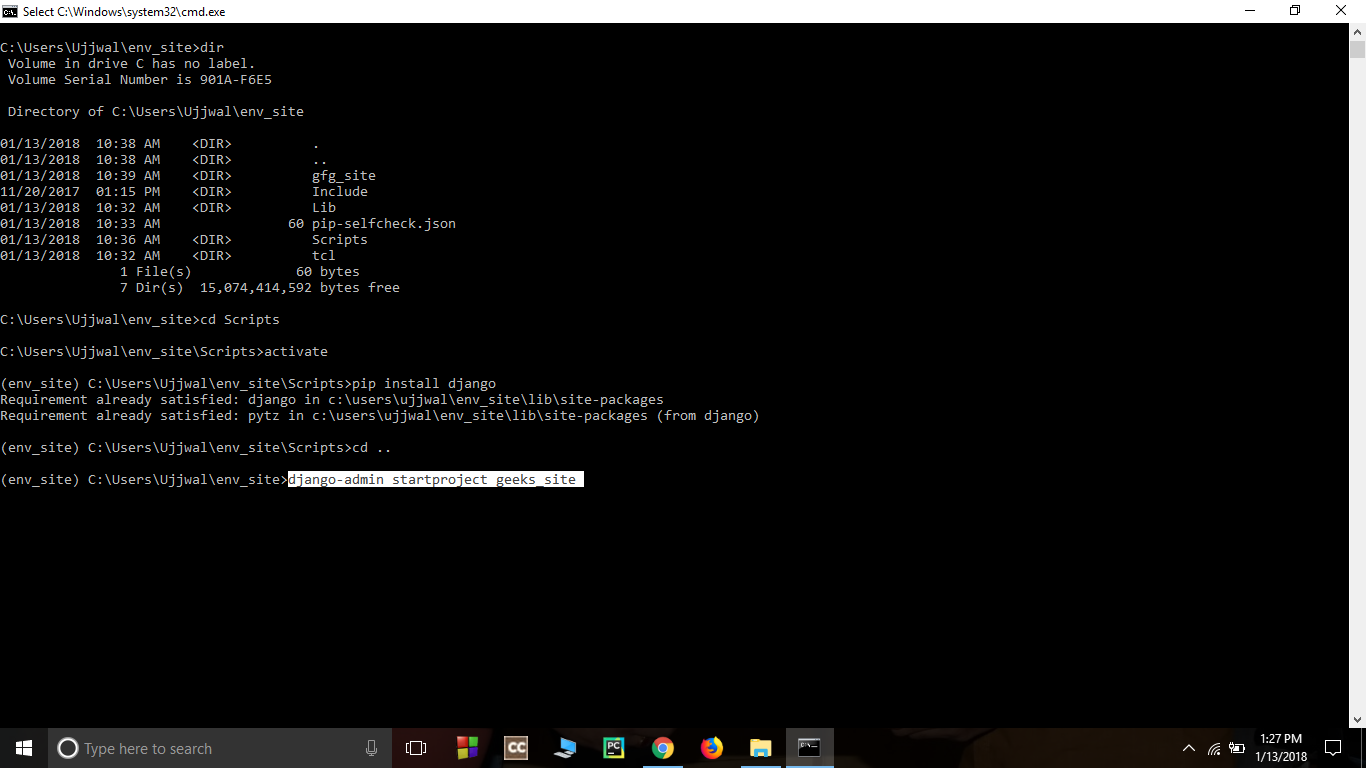
Return to the env\_site directory

cd ..

Figure 7.7 Set Directory

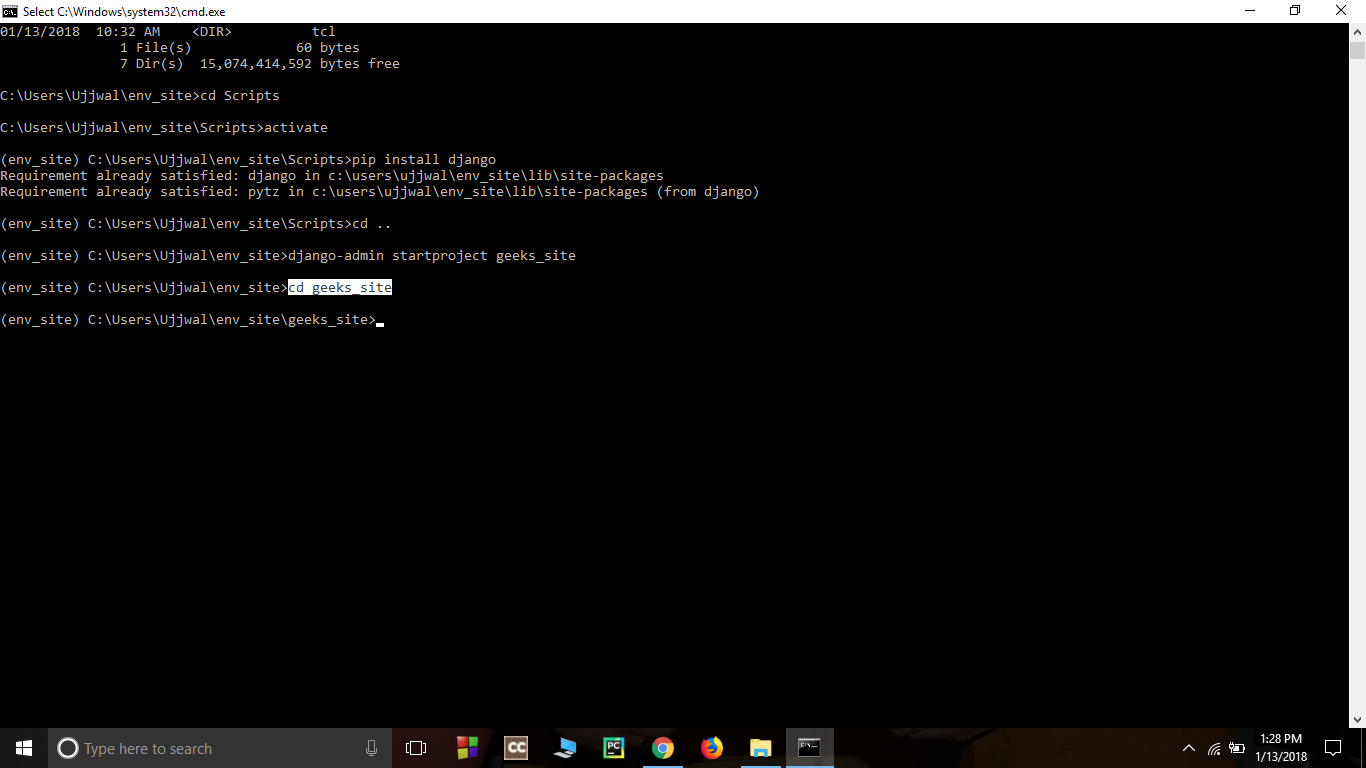
Start a project by following command

django-admin startproject geeks\_site

Figure 7.8 Start Project

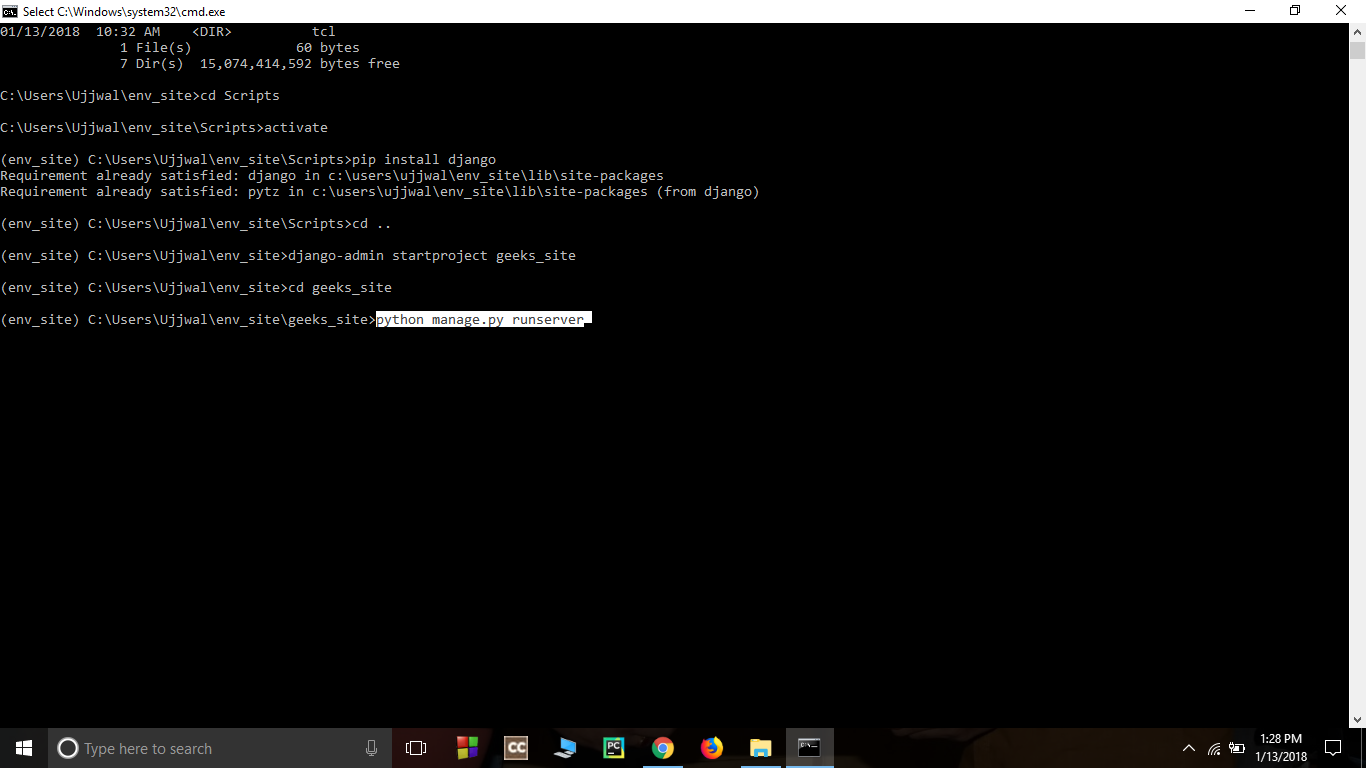
Change directory to geeks\_site

cd geeks\_site

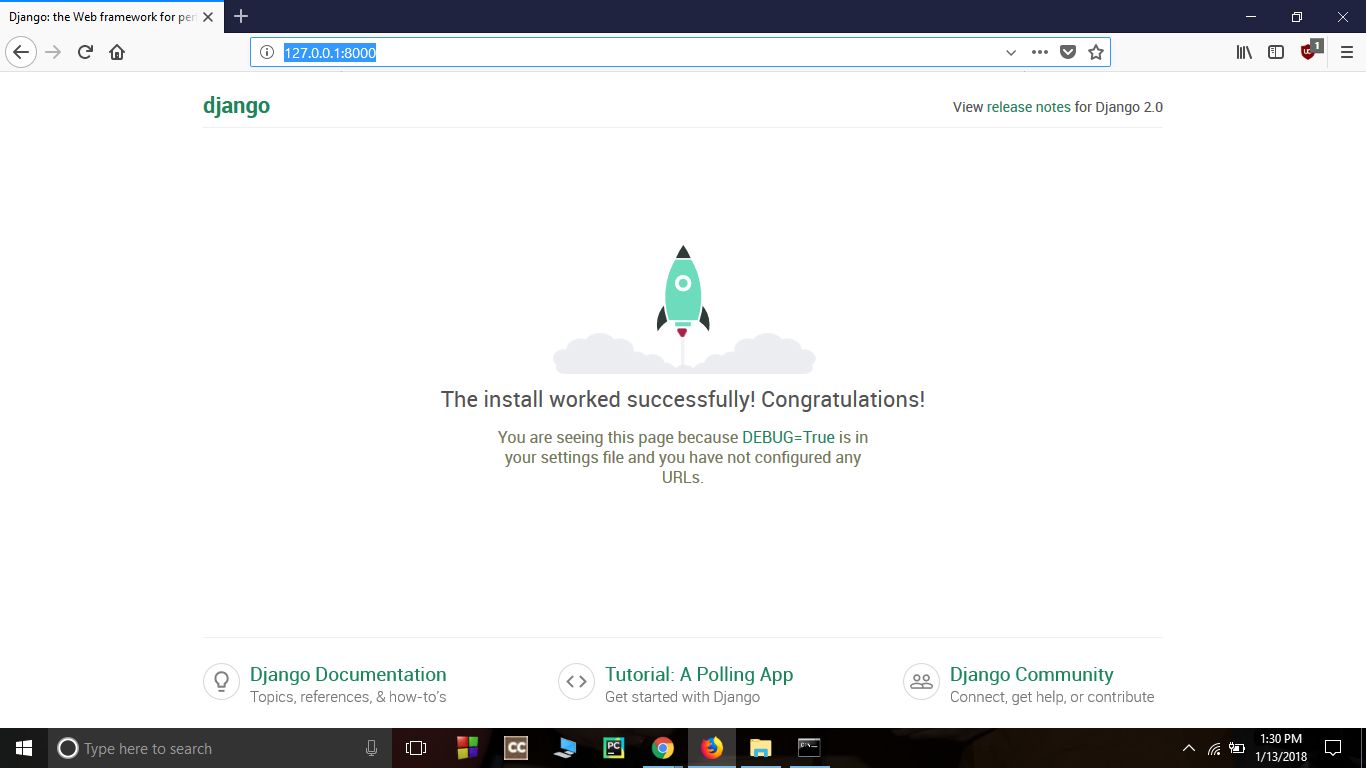
Figure 7.9 Change Directory

**Start the server-** Start the server by typing following command in cmd

python manage.py runserver

Figure 7.10 Start Server

To check whether the server is running or not go to the web browser and enter http://127.0.0.1:8000/ as url.

Figure 7.11 Url Open

**7.3 Create a Project**

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

$ django-admin startproject myproject

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

* myproject/
* manage.py
* myproject/
* \_\_init\_\_.py
* settings.py
* urls.py
* wsgi.py

**The Project Structure**

The “myproject” folder is just your project container, it actually contains two elements −

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

$ python manage.py help

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

* \_\_init\_\_.py − Just for python, treat this folder as a package.
* settings.py − As the name indicates, your project settings.
* urls.py − All links of your project and the function to call. A kind of ToC of your project.
* wsgi.py − If you need to deploy your project over WSGI.

**7.4 Create an Application**

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

$ python manage.py startapp myapp

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

* myapp/
* \_\_init\_\_.py
* admin.py
* models.py
* tests.py
* views.py
* \_\_init\_\_.py − Just to make sure python handles this folder as a package.
* admin.py − This file helps you make the app modifiable in the admin interface.
* models.py − This is where all the application models are stored.
* tests.py − This is where your unit tests are.
* views.py − This is where your application views are.

**Get the Project to Know About Your Application**

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

INSTALLED\_APPS = (

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'myapp',

)

**7.5 Admin Interface**

The Admin interface depends on the django.contrib module. To have it working you need to make sure some modules are importedintheINSTALLED\_APPSandMIDDLEWARE\_CLA SSEStuplesof the myproject/settings.py file.

To start the Admin Interface, we need to make sure we have configured a URL for our admin interface. Open the myproject/url.py and you should have something like −

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

from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',

# Examples:

# url(r'^$', 'myproject.views.home', name = 'home'),

# url(r'^blog/', include('blog.urls')),

url(r'^admin/', include(admin.site.urls)),

)

Now just run the server.

$ python manage.py runserver

And your admin interface is accessible at: **http://127.0.0.1:8000/admin/**

**7.6 MODEL**

A model is a class that represents a table or collection in our DB, and where every attribute of the class is a field of the table or collection. Models are defined in the app/models.py (in our example: myapp /models.py)

Creating a Model Following is a Dreamreal model created as an example −

from django.db import models

class Dreamreal(models.Model):

website = models.CharField(max\_length = 50)

mail = models.CharField(max\_length = 50)

name = models.CharField(max\_length = 50)

phonenumber = models.IntegerField()

class Meta:

db\_table = "dreamreal"

**7.7 GENERIC VIEWS**

In some cases, writing views, as we have seen earlier, is really heavy. Imagine you need a static page or a listing page. Django offers an easy way to set those simple views that are called generic views. Unlike classic views, generic views are classes not functions. Django offers a set of classes for generic views in django.views.generic, and every generic view is one of those classes or a class that inherits from one of them.

There are 10+ generic classes −

>>> import django.views.generic

>>> dir(django.views.generic)

**Django Views**

A view is a place where we put our business logic of the application. The view is a python function which is used to perform some business logic and return a response to the user. This response can be the HTML contents of a Web page, or a redirect, or a 404 error.All the view functions are created inside the views.py file of the Django app.

**Django View Simple Example**

//views.py

1. import datetime

2. # Create your views here.

3. from django.http import HttpResponse

4. def index(request):

5. now = datetime.datetime.now()

6. html = "<html><body><h3>Now time is %s.</h3></body></html>" % now

7.return HttpResponse(html) # rendering the template in HttpResponse

**Django URL Mapping**

Well, till here, we have learned to create a model, view, and template. Now, we will learn about the routing of applications. 65 Since Django is a web application framework, it gets user requests by URL locator and responds back. To handle URLs, the django.urls module is used by the framework.

Let's open the file urls.py of the project and see the what it looks like:

// urls.py

from django.contrib import admin

from django.urls import path

urlpatterns = [

path('admin/', admin.site.urls),

]

See, Django already has mentioned a URL here for the admin. The path function takes the first argument as a route of string or regex type.

**CHAPTER 8**

**MERN**

**8.1 Introduction** 

Today Developers around the world are making efforts to enhance user experience of using applications as well as to enhance the developer’s workflow of designing applications to deliver projects and rollout change requests under strict timeline. Stacks can be used to build web applications in the shortest span of time. The stacks used in web development are basically the response of software engineers to current demands. They have essentially adopted pre-existing frameworks (including JavaScript) to make their lives easier. While there are many, MEAN and MERN are just two of the popular stacks that have evolved out of JavaScript. Both stacks are made up of open source components and offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases. The common theme between the two is JavaScript and this is also the key benefit of using either stack. One can basically avoid any syntax errors or any confusion by just coding in one programming language, JavaScript. Another advantage of building web projects with MERN is the fact that one can benefit from its enhanced flexibility. In order to understand the MERN stack, we need to understand the four components that make up the MERN stack, namely – MongoDB, Express.js, React and Node.js.

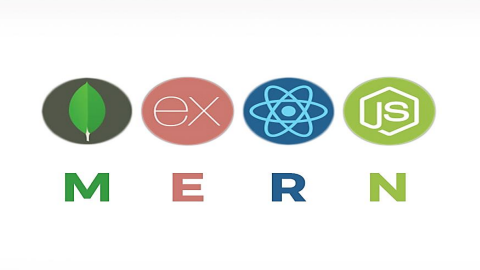


Figure 8.1 MERN

**8.2 MongoDB**

* MongoDB is a cross-platform document-oriented NoSQL database used for high volume data storage that provides high performance, high availability and easy scalability.
* MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in the application code, making data easy to work with.
* The data model available within MongoDB allows users to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
* MongoDB works on the concept of collections and documents. Each database contains collections which in turn contains documents. Each document can have a varying number of fields. The size and content of each document can also be different from each other.

**Key Components of MongoDB Architecture**

1. **\_id** – This is a 24-digit unique identifier field required in every MongoDB document in the collection. The \_id field is like the document's primary key. If the user creates a new document without an \_id field, MongoDB will automatically create the field.
2. **Collection -** Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Typically, all documents in a collection are of similar or related purpose.
3. **Document -** A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.
4. **Database -** Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
5. **Field -** A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.

**8.3 Express.js**

• Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is an open source framework developed and maintained by the Node.js foundation.

• Express provides us the tools that are required to build our app, be it single-page, multi-page or hybrid web applications. It is flexible as there are numerous modules available on npm(Node Package Manager), which can be directly plugged into Express.

• Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.

• Pug (earlier known as Jade) is a terse language for writing HTML templates. It produces HTML, supports dynamic code and code reusability (DRY). It is one of the most popular template languages used with Express.

• Express can be thought of as a layer built on the top of the Node.js that helps manage a server and routes. It allows users to setup middleware to respond to HTTP Requests and defines a routing table which is used to perform different actions based on HTTP method and URL.

• Express allows to dynamically render HTML Pages based on passing arguments to templates.

• Express is asynchronous and single threaded and performs I/O operations quickly.

**Why use Express?**

• Ultra-fast I/O.

• Asynchronous and single threaded.

• MVC like structure.

• Robust API makes routing easy.

**8.4 React**

• ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

• A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses a virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

• Instead of using regular JavaScript, React codes are written in something called JSX (JavaScript Syntax Extension). JSX is basically a syntax extension of regular JavaScript and is used to create React elements. These elements are then rendered to the React DOM. JSX is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript.

**Why use React?**

• Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.

• Can be used on client and server side as well as with other frameworks.

• Component and data patterns improve readability, which helps to maintain larger apps.

**8.5 Node.js**

• Node.js is a very powerful JavaScript-based platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.

• Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009.

• Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

• Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

**Features of Node.js**

1. Extremely fast: Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.

2. I/O is Asynchronous and Event Driven: All APIs of Node.js library are asynchronous i.e. non-blocking. So, a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.

3. Single threaded: Node.js follows a single threaded model with event looping.

4. Highly Scalable: Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.

5. No buffering: Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.

6. Open source: Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js application.

**CHAPTER 9**

**REAL TIME CHATTING APPLICATION**



**9.1** **INTRODUCTION**

Chat Room has been the most basic step toward creating real-time and live projects. The chat page that user we will create will be a simple HTML boilerplate with a simple h1 text with the name of the current user and a link to log out to the user who is just logged in. You may need to comment on the line until we create auth system for this. This ensures that when two users are chatting, one can log out and it will not affect the other

**Product Definition**

Channels are the python project which was created to extend the ability of Django to the next level. We were working in standard Django which did not support asynchronous and channels and connection via Web Sockets to create real-time applications. Channels extend the ability of Django beyond HTTP and make it work with Web Sockets, chat protocols, IoT protocols, and more. It is built on ASGI support which stands for Asynchronous Server Gateway Interface. ASGI is the successor of WSGI which provides an interface between async and python. Channels provide the functionality of ASGI, by extending WSGI to it, and it provides ASGI support with WSGI. Channels also bundle the event-driven architecture with the channel layers, a system that allows you to easily communicate between processes and separate your project into different processes.

* It’s a Desktop based application.
* It is based on Python
* It provides a user-friendly environment.
* To make it accessible with many features enabled.
* To make it user friendly
* To enable security, privacy and anonymity
* To make it with a simple GUI

**8.2 Problem Statement**

* Before years ago, people were liking to interact each other and sharing the thoughts, but nowadays trend is changed, youngsters are not believed in real relationship, but they liking to Live in Virtual world, due to the expansion of growth of internet facilities rapidly, So nowadays Chat grabs the internet market.
* This project is to create a chat application with a server and clients to enable the client to chat with many other client int the same common chat groups.
* This project is to simulate the multicast chatting. In the case of multicasting when a message is sent to group of clients, then only a single message is sent to the router

**Innovative Ideas of Project**

* GUI: Easy to use GUI (Graphical User Interface), hence any user with minimal knowledge of operating a system can use the software.
* Unlimited clients: “N” number of users can be connected without any performance degradation of the server.

**9.3 Project Objective**

* There is two-way communication between different client and server.
* The purpose of this project is to implement a chat application that will allow user with an internet connection to engage in private and public conversation.
* The development of this project centred on the development is a message protocol that would allow the application to properly log in users, send message and perform system maintenance.

**9.4 Scope of The Project**

* There is always a room for improvements in any software package, however good and efficient it may be done. But the most important thing should be flexible to accept further modification right now we are just dealing with text communication. In future this software may be extended to include features like FILE TRANSFER and VOICE CHAT, etc.

**Feasibility Analysis**

Multiple functionalities along with privacy and security fulfill the user requirements. Various

newly added functions such as sending file via socket have been added to enhance

customer experience and provide necessary utilities to the user which makes it feasible in

the long term of the software.

**9.5 SYSTEM REQUIREMENT**

**9.5.1 Programming Languages and Development Tools**

* Python 3.9
* JavaScript
* HTML
* CSS
* Django

**9.5.2 System Requirement Specifications**

**Client-Side**

**Processor:** Intel i5-7200U,2.5Ghz

**Operating system:** Windows 11

**Memory:** 8GB RAMHard

**Disk space:** 1TB HDD+256 GB SSD

**Software version:** Python 3.9

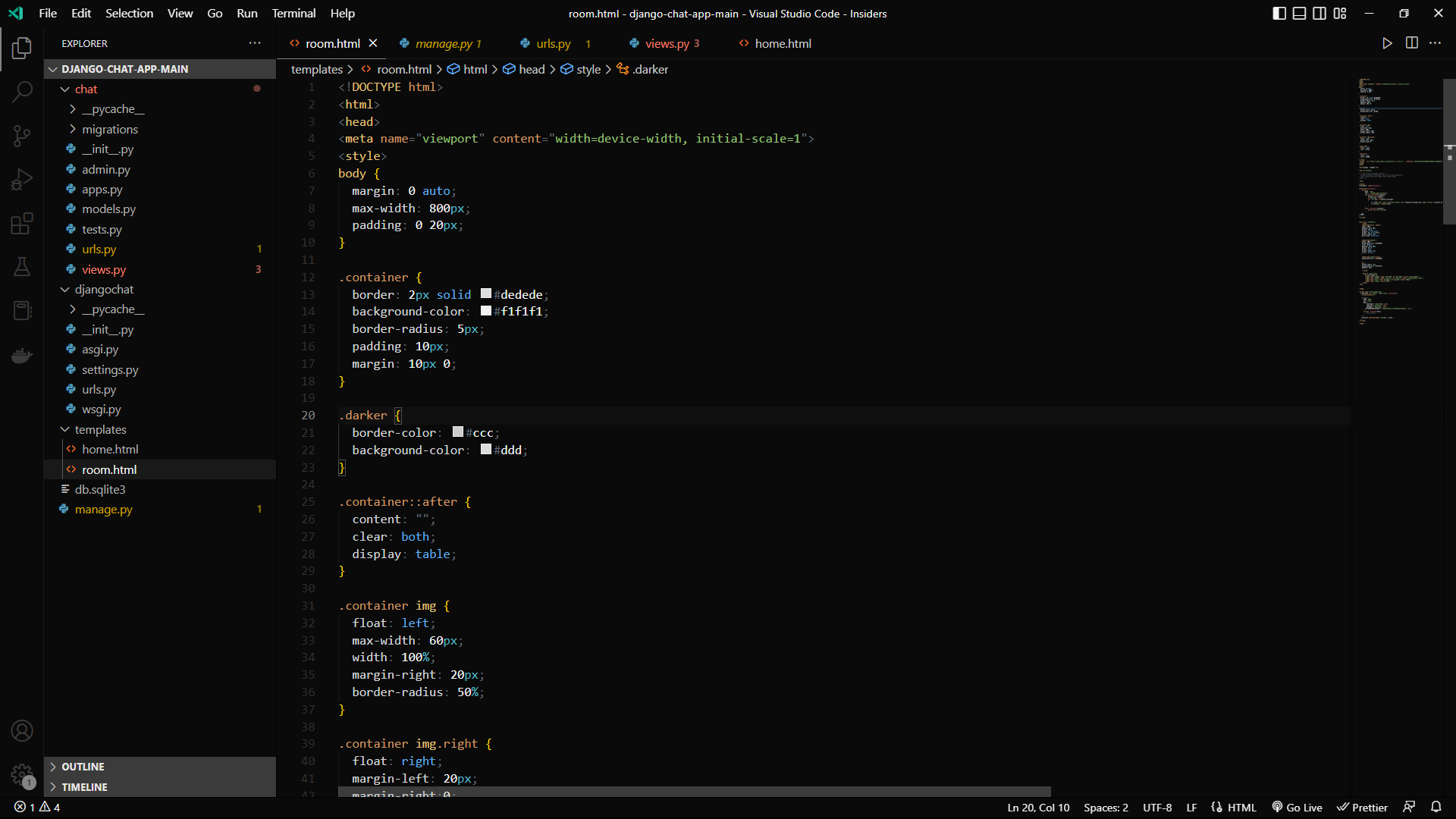
**9.6 WORKFLOW**

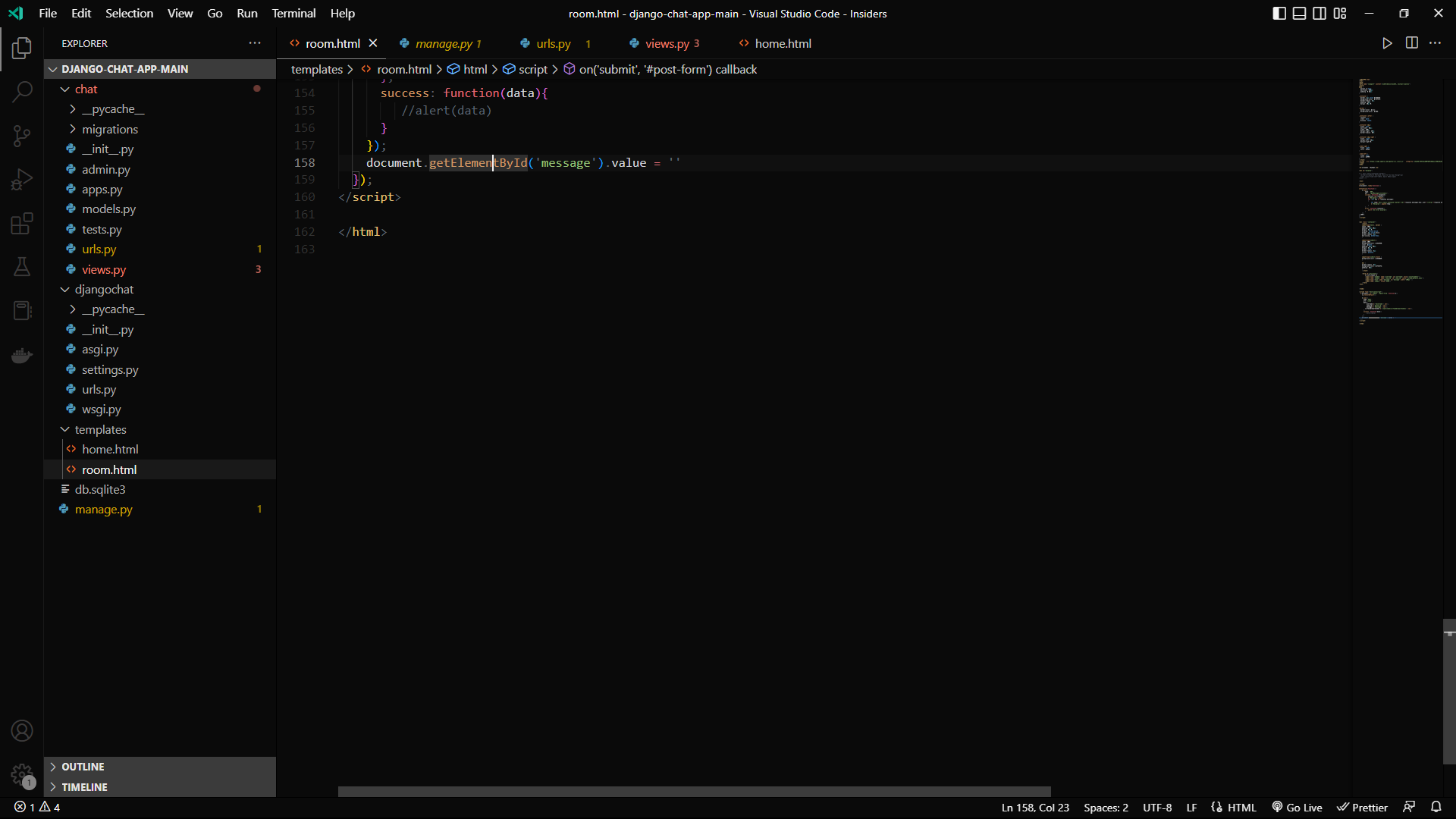
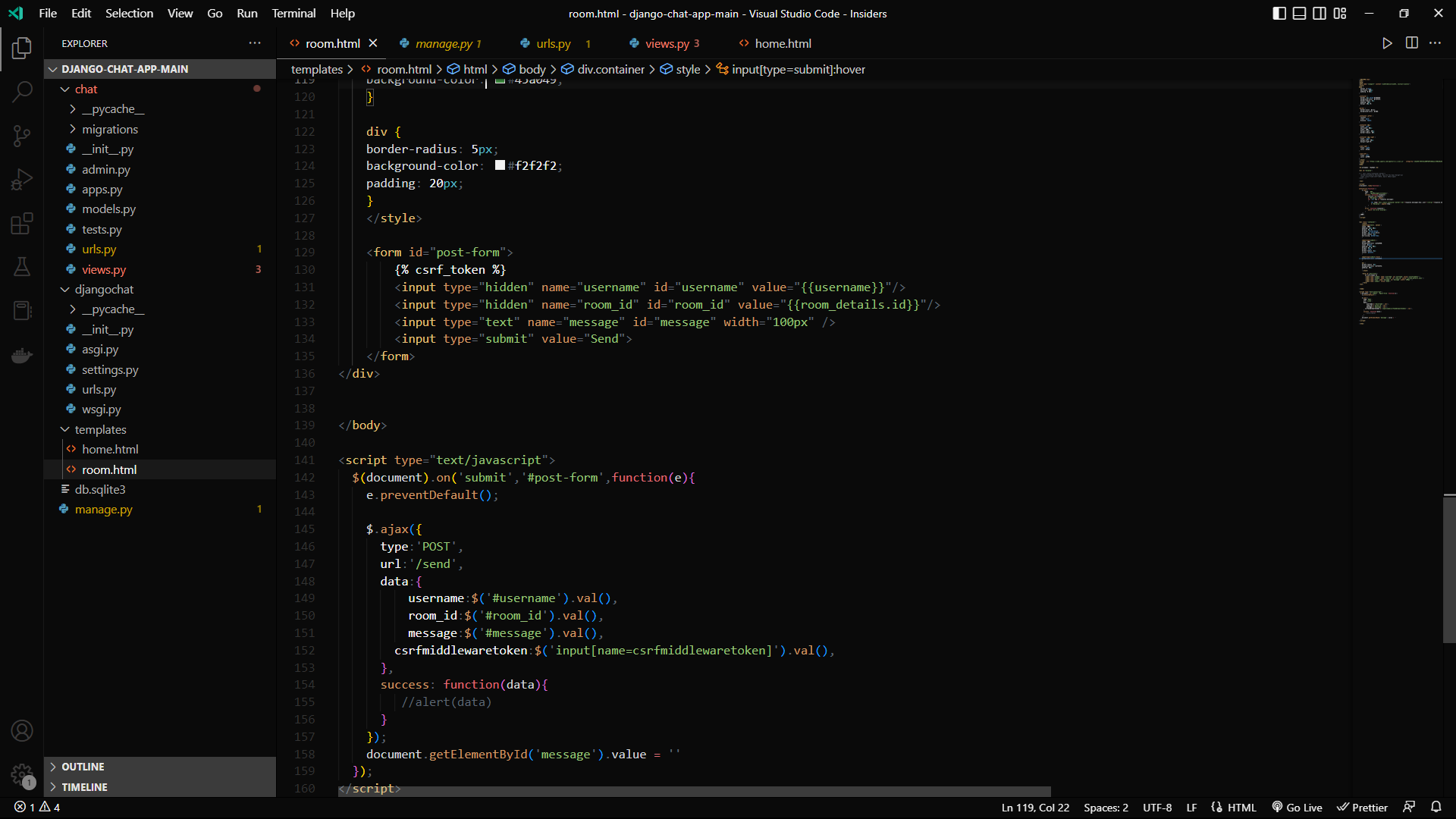
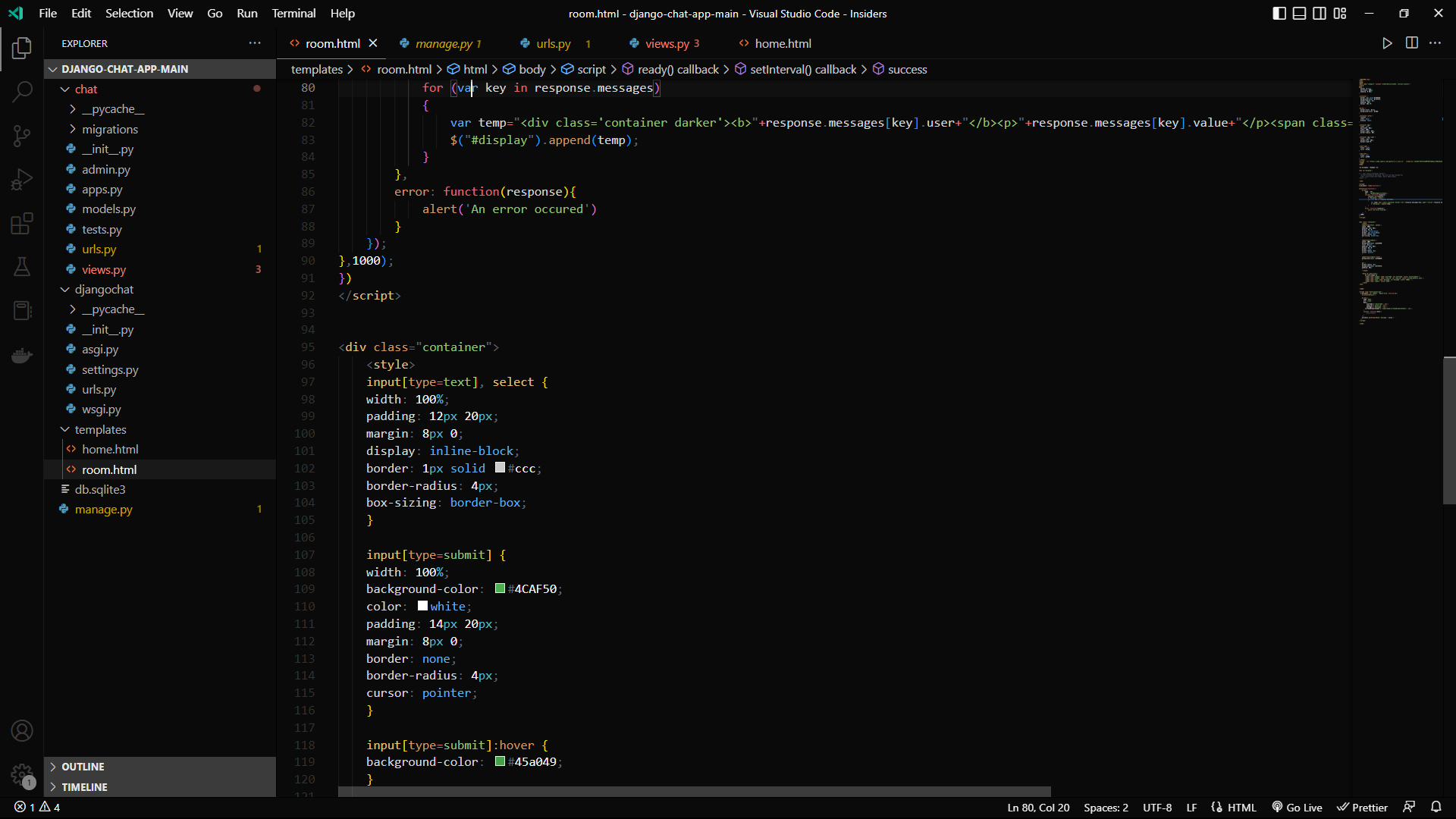
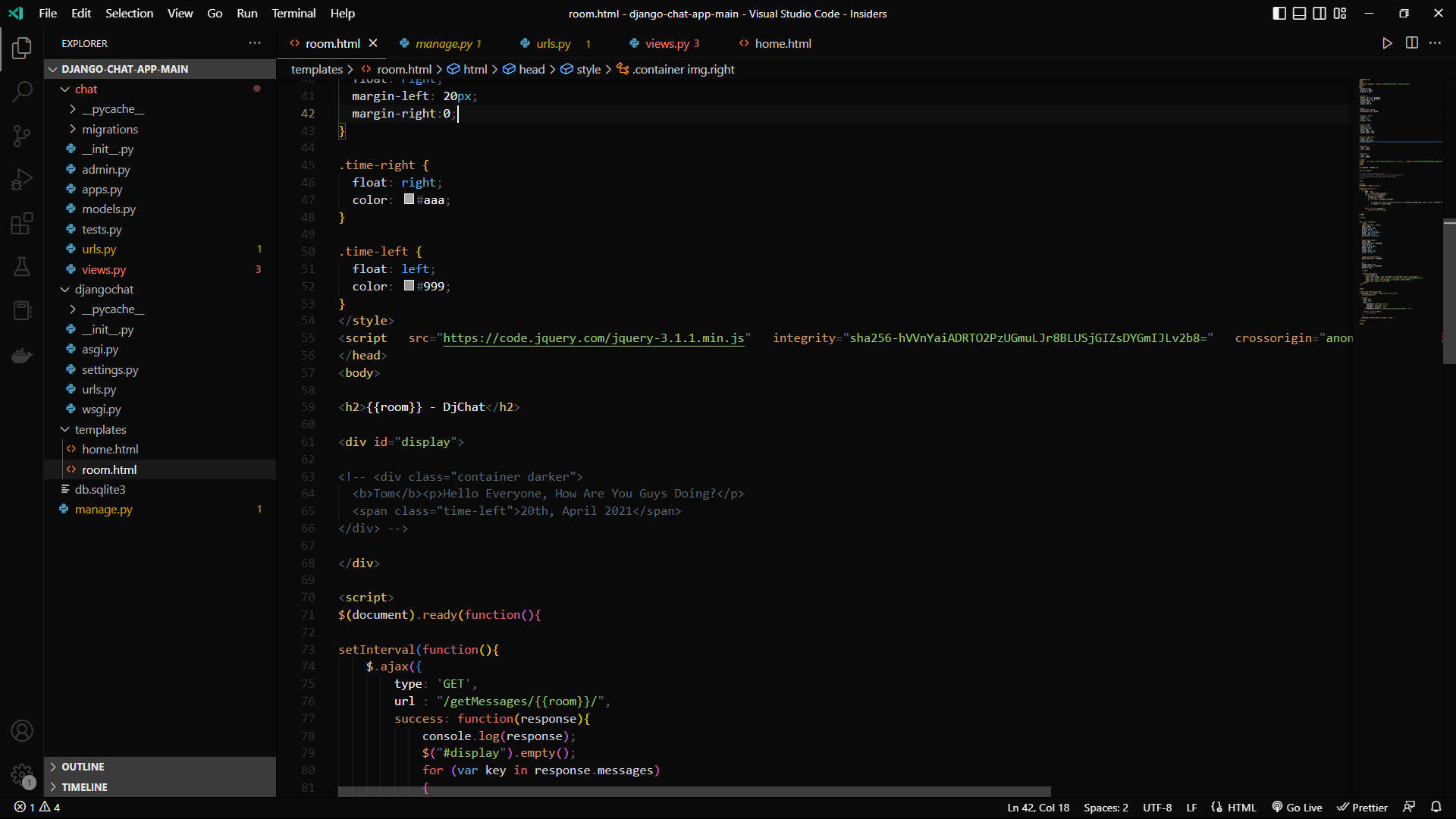
Diagram

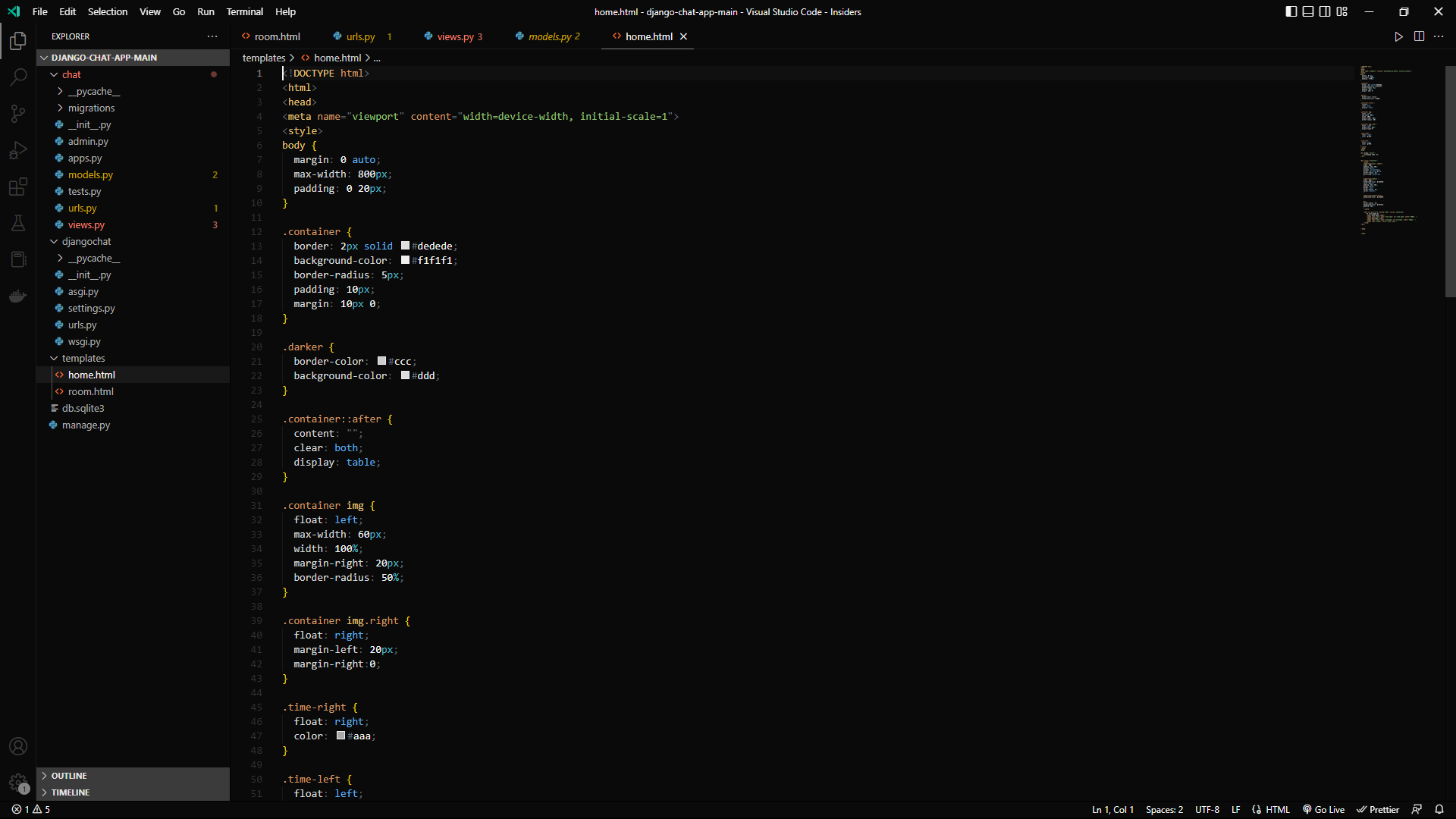
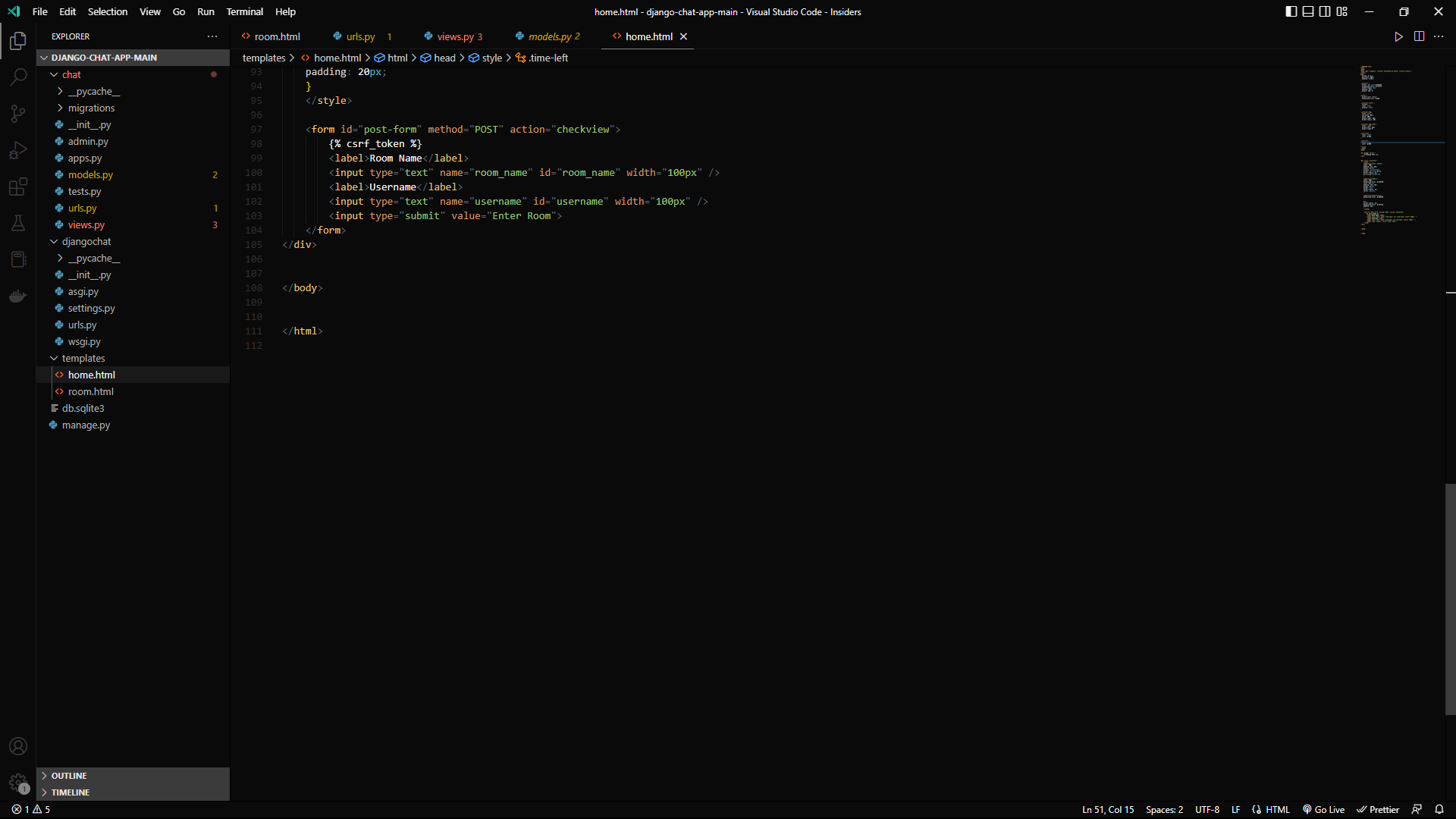
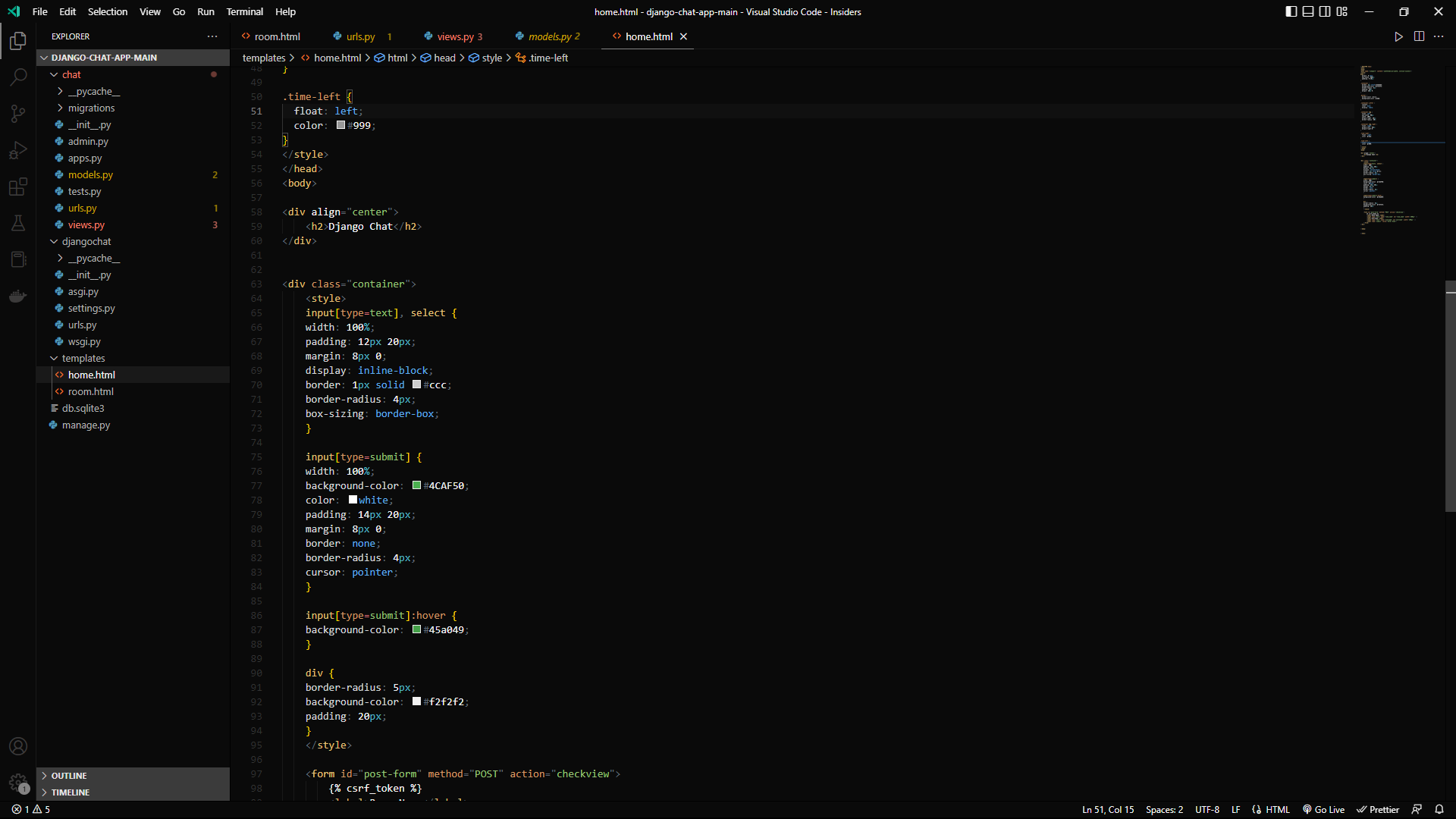
Description automatically generated

Figure 9.1 Flowchart Diagram

**9.7 SNAPSHOTS**



Figure 9.2 ROOM HTML CODE

  Figure 9.2 HOME HTML CODE

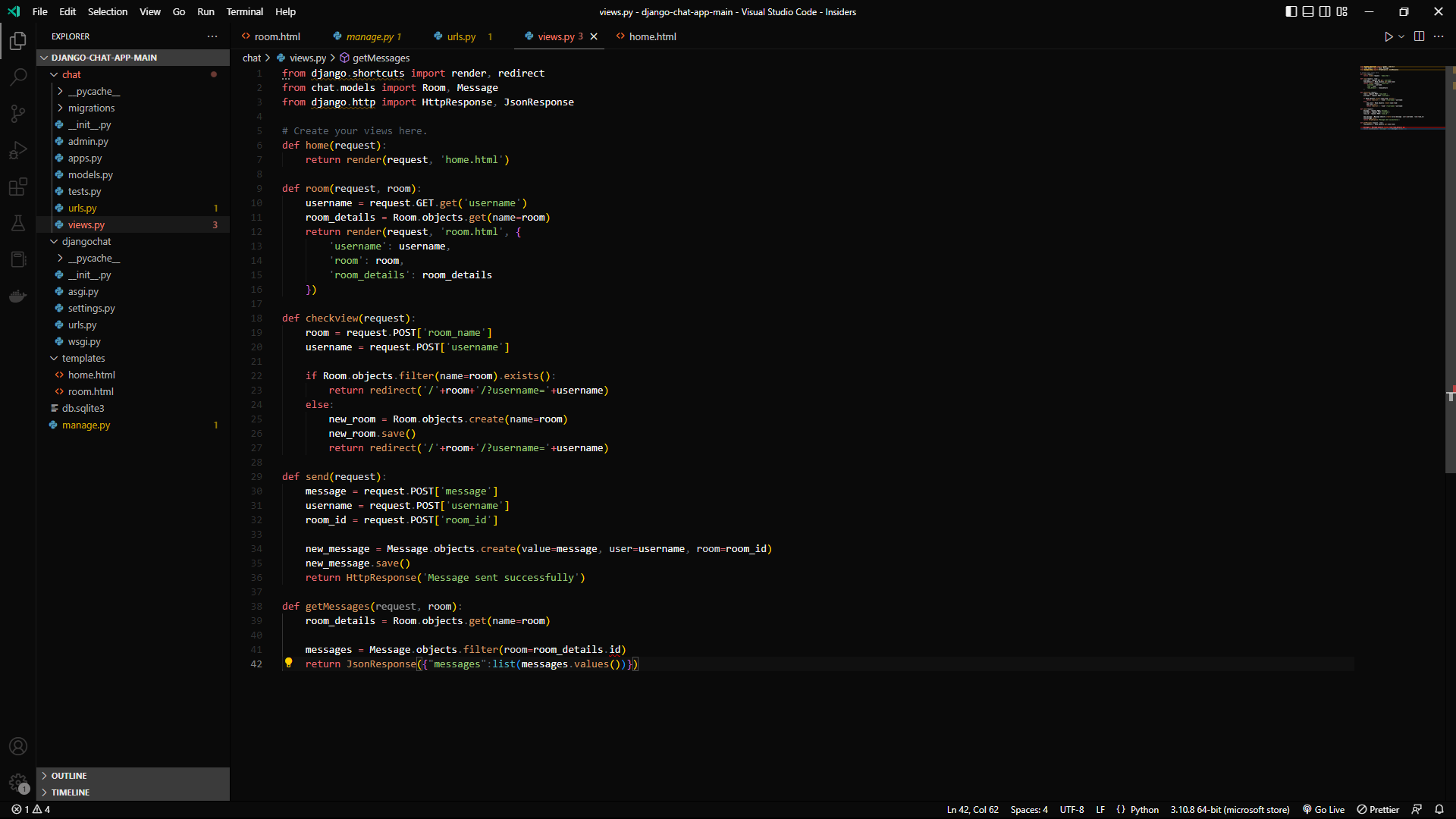
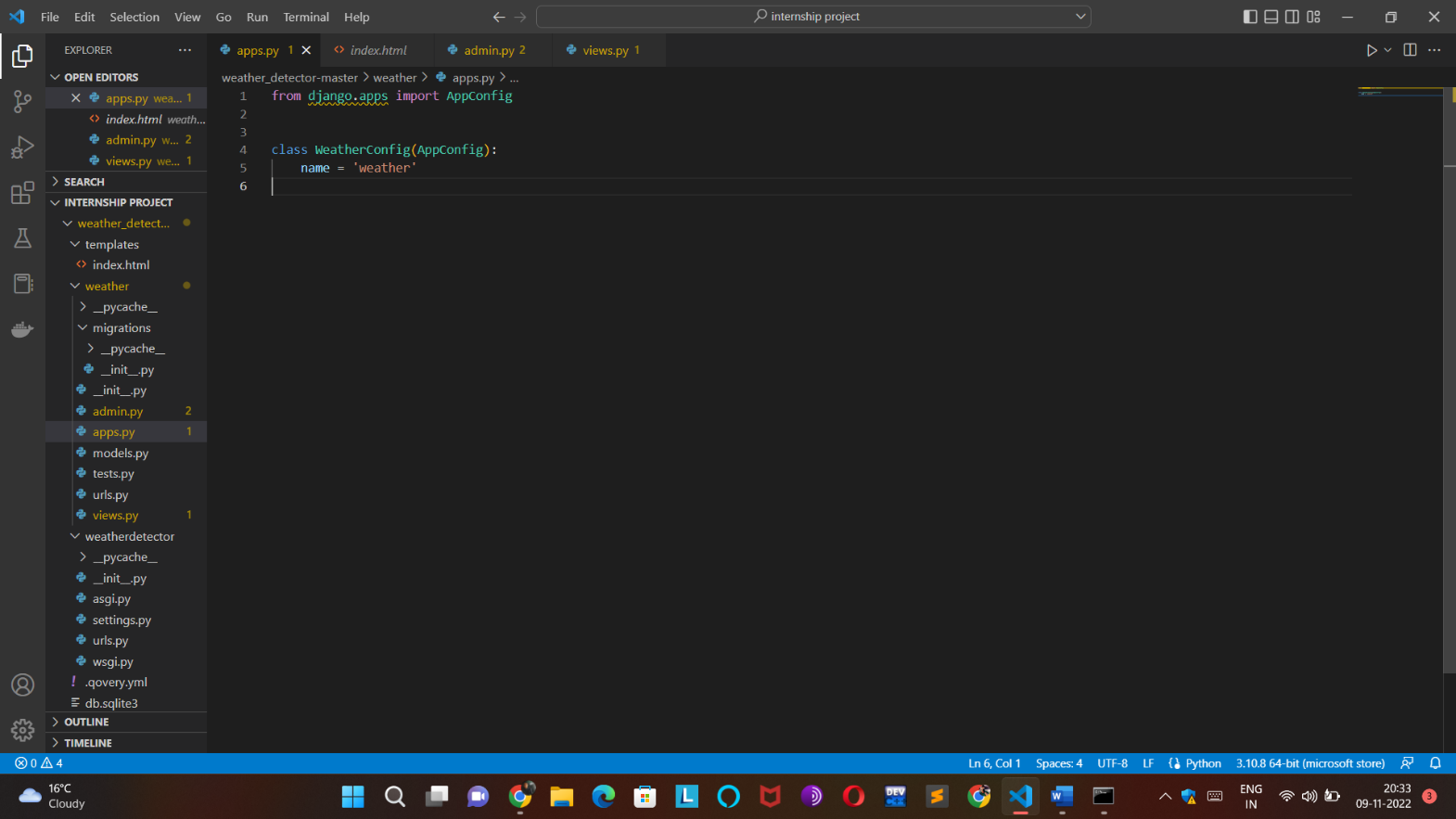


Figure 9.3 python file (view)

Figure 9.4 Python Code

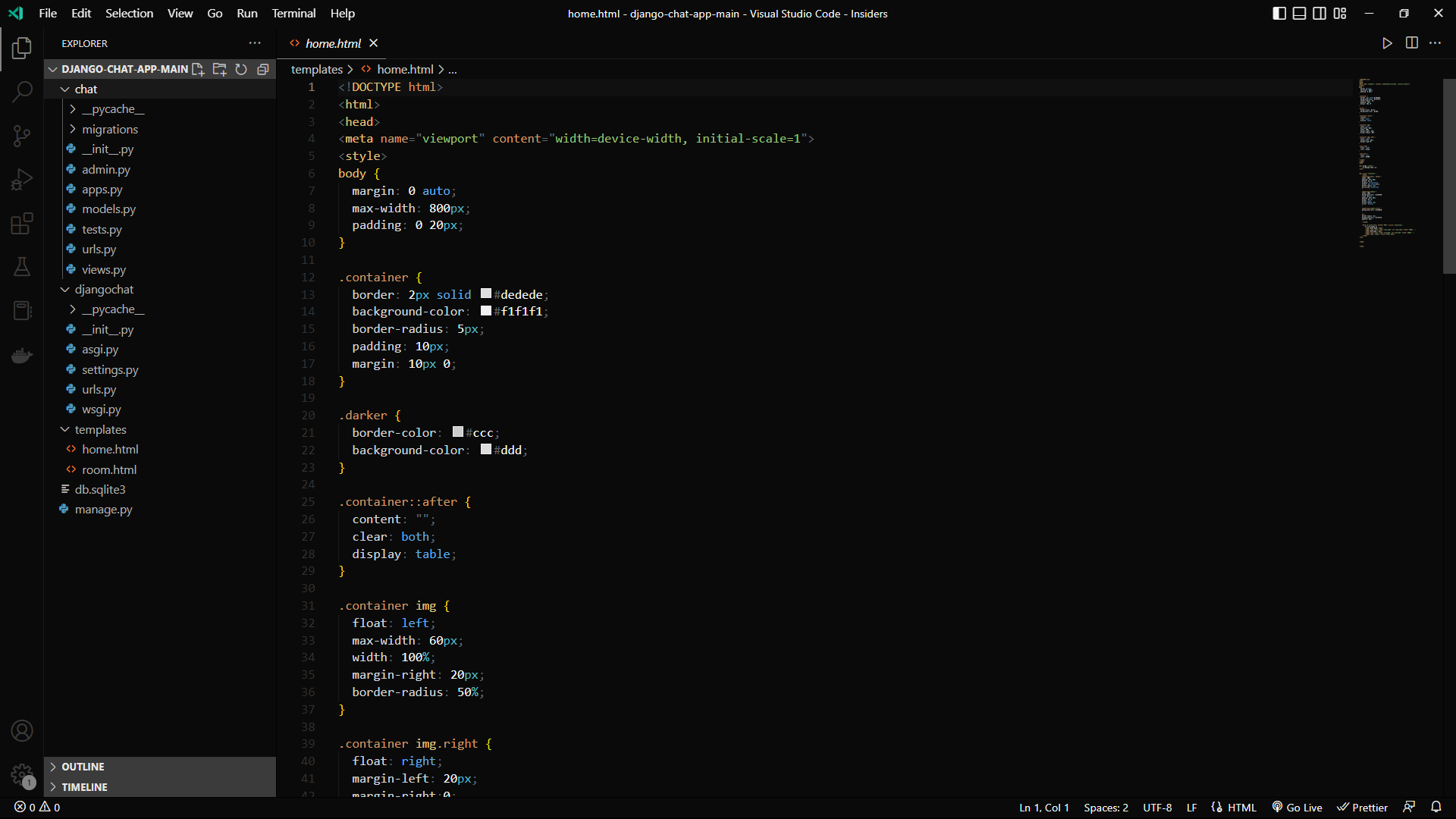
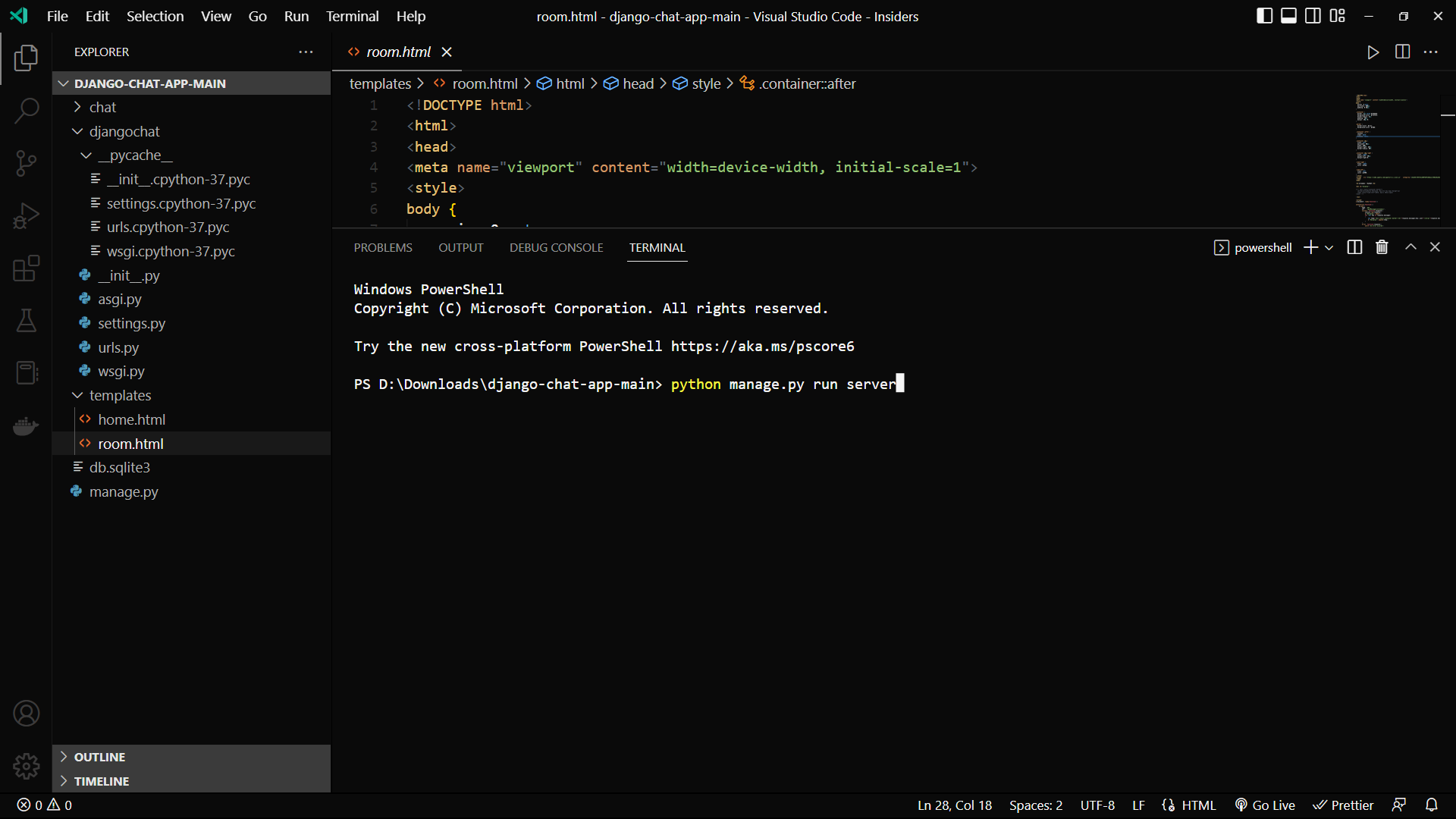
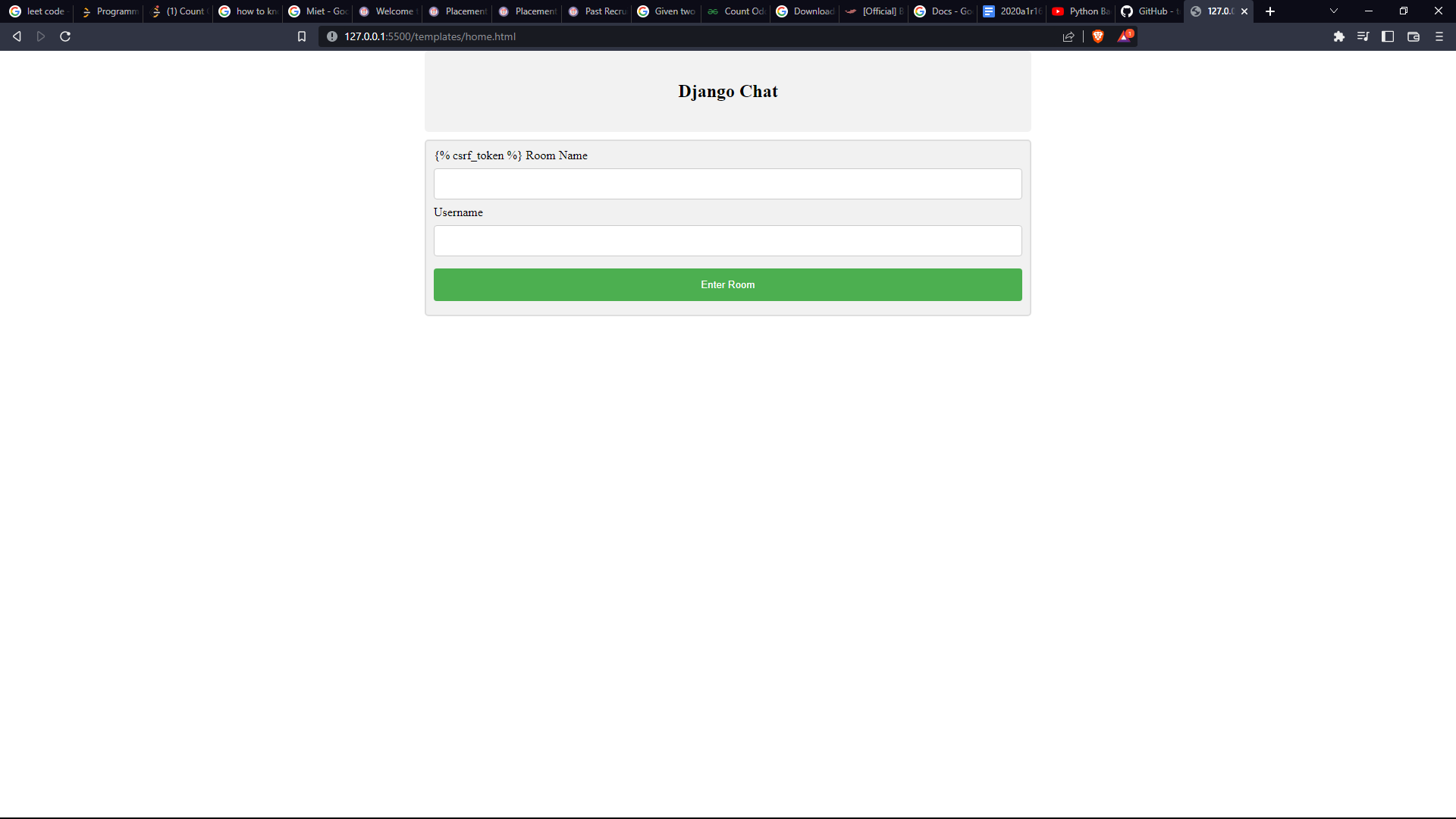


Figure 9.5 CHAT-APP PYTHON FILES



Figure 9.8 Execution

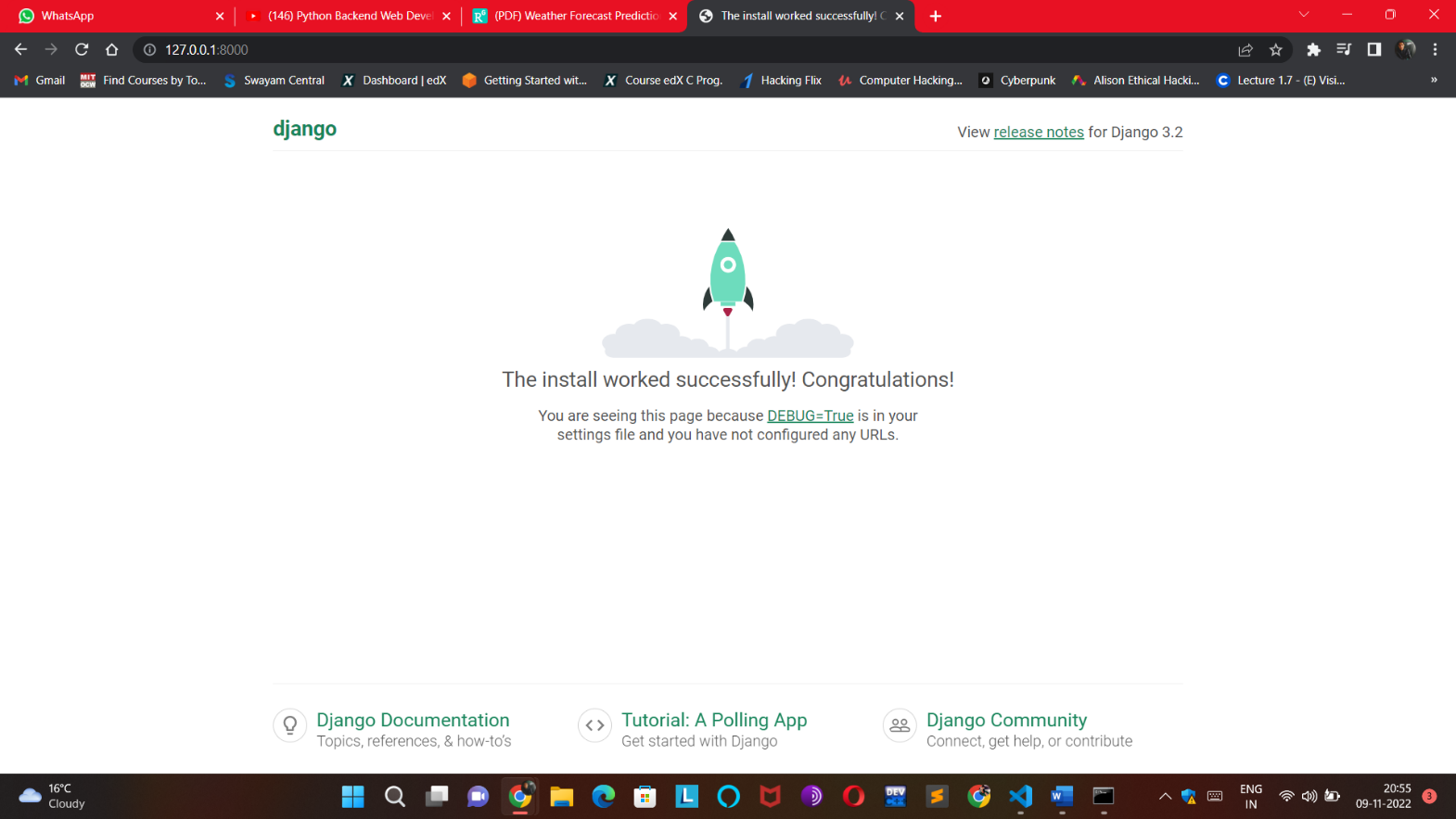


Figure 9.9 Django Workspace

Figure 9.10 Registration

Graphical user interface, application

Description automatically generated

Figure 9.11 working

**9.8 CONCLUSION**

This paper works with mix of Naïve Bayes and Chi Square algorithm to predict weather condition. The constant information i.e. time-series data is assembled and analysis is performed on this dataset utilizing an interface named Weather Prediction System, developed utilizing Java using Eclipse tools. This framework arranges the given information into various classifications and furthermore predicts the risk of the weather prediction of obscure example is given as an input. The system can be filled in as training tool for Meteorology Students. This methodology can decide the non-linear relationship that exists between the historical data (temperature, wind speed, humidity, and so forth.,) provided to the system during the training phase and on that premise, make a prediction of what the weather would be in future. The Future work of this project is to incorporate more attribute of weather condition to predict and to work with other classification algorithm to become more accurate in prediction

# REFERENCES

1. MongoDB: <https://docs.mongodb.com/ecosystem/drivers/> ,<https://www.guru99.com/what-is-mongodb.html>.
2. ExpressJS: <https://expressjs.com/en/guide/routing.html>, <https://www.javatpoint.com/expressjs-tutorial>.
3. Npm: <https://www.npmjs.com/>
4. ReactJS: <https://reactjs.org/docs/getting-started.html>,

<https://www.javatpoint.com/reactjs-tutorial>,

<https://www.tutorialspoint.com/reactjs/reactjs_overview.html>.

1. NodeJS: <https://nodejs.org/en/docs/>, <https://www.javatpoint.com/nodejs-tutorial> .
2. WebSocket: <https://yellow.systems/blog/guide-to-the-chat-architecture>
3. REST API: <https://www.toptal.com/nodejs/secure-rest-api-in-nodejs>