Anjuman-I-Islam’s

M.H. Saboo Siddik Polytechnic



Subject name: Programming with python (PWP) - 22616

Department: COMPUTER ENGINEERING

SemeSter: SiXTH

micro project title: Chat Application (PyCHat)

Prepared by:

Year: 2023-24

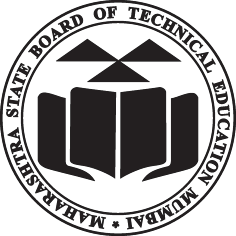
Names of Team Members with Roll Nos.

1. Abdurrahman Qureshi - 210451

2. Oaish Qazi - 210455

3. Shaikh Mohammed Hussain - 220486

Under the guidance of: Prof. Zaibunnisa Malik



**Maharashtra State**

**Board of Technical Education**

**Certificate**

This is to certify that Mr. Abdurrahman Qureshi of Sixth Semester of Diploma in Computer Engineering of Institute M.H. Saboo Siddik Polytechnic has successfully completed Micro-project work in subject Programming With Python (22616) for the academic year 2023- 2024 as prescribed in the I-Scheme Curriculum.

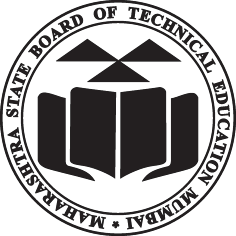
Place: …………………… Enrollment no: .…………………..

Date: …………………… Exam seat no: …………………....

|  |  |  |
| --- | --- | --- |
| **Signature** | **Signature** | **Signature** |
| **Project Guide** | **H. O. D** | **Principal** |

Seal of

Institute



**Maharashtra State**

**Board of Technical Education**

**Certificate**

This is to certify that Mr. Oaish Qazi of Sixth Semester of Diploma in Computer Engineering of Institute M.H. Saboo Siddik Polytechnic has successfully completed Micro-project work in subject Programming With Python (22616) for the academic year 2023-2024 as prescribed in the I-Scheme Curriculum.

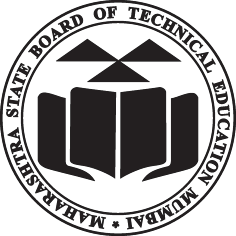
Place: …………………… Enrollment no: .…………………..

Date: …………………… Exam seat no: …………………....

|  |  |  |
| --- | --- | --- |
| **Signature** | **Signature** | **Signature** |
| **Project Guide** | **H. O. D** | **Principal** |

Seal of

Institute



**Maharashtra State**

**Board of Technical Education**

**Certificate**

This is to certify that Mr. Shaikh Mohammed Hussain of Sixth Semester of Diploma in Computer Engineering of Institute M.H. Saboo Siddik Polytechnic has successfully completed Micro-project work in subject Programming With Python (22616) for the academic year 2023-2024 as prescribed in the I-Scheme Curriculum.

Place: …………………… Enrollment no: .…………………..

Date: …………………… Exam seat no: …………………....

|  |  |  |
| --- | --- | --- |
| **Signature** | **Signature** | **Signature** |
| **Project Guide** | **H. O. D** | **Principal** |

Seal of

Institute

Acknowledgment

We wish to express our profound gratitude to our guide Ms. Zaibunnisa Malik who guided us endlessly in the framing and completion of the micro project. She guided us on all the main points in that micro project. We are indebted to his constant encouragement, cooperation, and help. It was his enthusiastic support that helped us in overcoming various obstacles in the micro-project.

We are also thankful to our Principal, HOD, faculty members and classmates of Computer Engineering department for extending their support and motivation in the completion of this micro-project.

Names of Team Members with Roll Nos.

1. Abdurrahman Qureshi - 210454

2. Oaish Qazi - 210455

3. Shaikh Mohammed Hussain - 220486

***Micro-Project Proposal***

**Chat Application**

**Annexure-I**

**1.0 Aims/Benefits of the Micro-Project**

* The aim of this micro-project is to create a real-time chat application using web technologies, with a specific focus on learning and applying Django for server-side programming. By undertaking this project, you can gain valuable hands-on experience in developing interactive web applications. The benefits of this endeavor include enhancing your understanding of Python-based web development, improving your proficiency in Django, and web application architecture, and achieving the creation of a functional chat application for personal or educational use. Additionally, this project presents the opportunity to potentially publish your application online, serving as a showcase of your web development skills.

**2.0 Course Outcomes Addressed**

* Develop python program to demonstrate use of Operators
* Perform operations on data structures in Python.
* Develop functions for given problem.
* Design classes for given problem.
* Handle exceptions.

**3.0 Proposed Methodology**

* Discussion of topic with guide and group members.
* Dividing the work and Gathering information about the project.
* Submission of project proposal. (Annexure I).
* Collection and Analysis of data / information from various sources.
* Set up the development environment (e.g., PyCharm, Django server).
* Design the user interface for the chat application using HTML/CSS templates.
* Implement Django routes to handle user registration, login, and chat functionality.
* Utilize Django-SocketIO for real-time communication between users.
* Store chat messages in a database (e.g., SQLite, MySQL) for persistence.
* Apply security measures (e.g., CSRF protection, password hashing) to protect against common web vulnerabilities.
* Test the Project for Bugs and Errors.
* Preparation of the project report (Annexure II).
* Microproject Submission and Viva.

**4.0 Action Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Week** | **Details of activity** | **Planned**  **Start date** | **Planned**  **Finish date** | **Name of**  **Responsible Team Members** |
| 1 | 1 & 2 | Discussion and finalization of the |  |  | All |
| 2 | 3 & 4 | Dividing the work among group members. |  |  | All |
| 3 | 5 | Submission of micro project proposal  (Annexure I) |  |  | All |
| 4 | 6 | Collection of the information on the  Topic. |  |  | All |
| 6 | 7 | Collection of all relevant content / materials for the execution of the project. |  |  | All |
| 7 | 8 & 9 | Execution of collected data and preparing layout of the web app and building web app and websockets. |  |  | Oaish / Abdurrahman |
| 8 | 10 | Integration of frontend with backend |  |  | Oaish / Abdurrahman |
| 9 | 11 | Testing the project for bugs and errors |  |  | Oaish / Abdurrahman |
| 10 | 12 | Preparation of the project report  (Annexure II) |  |  | All |
| 11 | 13 | Microproject Submission and Viva |  |  | All |

**5.0 Resources Required**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Name of Resource/material** | **Specifications** | **Quantity** | **Remarks** |
| 1 | Software |  | 1 of Each |  |
| Git, Firefox, Jetbrains  Pycharm |  |
| Pycharm |  |
|  |  |
| 2 | CASE Tools | Pycharm, Django, VS Code, Notepad | 1 of Each |  |
| 3 | Browser | Opera GX, Firefox, Chrome, Edge | 1 of Each |  |

**Names of Team Members with Roll Nos.**

1. Abdurrahman Qureshi - 210454

2. Oaish Qazi - 210455

3. Shaikh Mohammed Hussain - 220486

**(To be approved by the concerned teacher)**

\*\*\*\*\*\*\*\*\*\*\*

**Micro-Project Report**

1. **Rationale**
2. **Chat Application**

With the increasing popularity of online communication and social networking, there is a growing demand for real-time chat applications. Building a chat application using Django REST Framework provides an opportunity to leverage Django's powerful features for web development, including authentication, authorization, and database management. Additionally, using RESTful APIs allows for seamless integration with frontend frameworks and ensures scalability and maintainability of the application.

**2.0 Aims/Benefits of the Micro-Project:**

The aim of this micro-project is to create a real-time chat application using web technologies, with a specific focus on learning and applying Django for server-side programming. By undertaking this project, you can gain valuable hands-on experience in developing interactive web applications. The benefits of this endeavor include enhancing your understanding of Python-based web development, improving your proficiency in Django, and web application architecture, and achieving the creation of a functional chat application for personal or educational use. Additionally, this project presents the opportunity to potentially publish your application online, serving as a showcase of your web development skills

1. **Course Outcomes Achieved**

* Develop python program to demonstrate use of Operators
* Perform operations on data structures in Python.
* Develop functions for given problem.
* Design classes for given problem.
* Handle exceptions.

1. **Literature Review**

• Research existing chat applications and their architectures.

• Study Django and Jinja documentation and tutorials.

• Review best practices for securing web applications.

• Explore WebSocket communication in web development.

• Learn about database integration for web applications (e.g., SQLAlchemy).

1. **Actual Methodology Followed**

* Discussed the topic with guide and group members.
* Divided the work and Gathered information about the project.
* Submitted project proposal. (Annexure I).
* Collected and Analyzed data / information from various sources.
* The development environment set up (e.g., PyCharm, Django server).
* Designed the user interface for the chat application using HTML/CSS templates.
* Implemented Django routes to handle user registration, login, and chat functionality.
* Utilized Django-SocketIO for real-time communication between users.
* Stored chat messages in a database (e.g., SQLite, MySQL) for persistence.
* Tested the Project for Bugs and Errors.
* Prepared the project report (Annexure II).
* Microproject Submitted with Viva.

**6.0 Actual Resources Used** (Mention the actual resources used).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Name of Resource/material** | **Specifications** | **Quantity** | **Remarks** |
| 1 | Software | Pycharm, FireFox, | 1 |  |
| 2 | Websites | web.whatsapp.com, youtube.com, geeksforgeeks.com | 1 |  |

**7.0 Outputs of the Micro-Projects**

**➢ Features of python:**

* **Free and Open Source.**
* **Easy to code.**
* **Easy to Read.**
* **Object-Oriented Language.**
* **GUI Programming Support.**
* **High-Level Language.**
* **Large Community Support.**
* **Easy to Debug.**
* **Python is a Portable language.**
* **Python is an Integrated language.**
* **Interpreted Language.**
* **Large Standard Library.**
* **Dynamically Typed Language.**
* **Frontend and backend development.**
* **Allocating Memory Dynamically.**

**➢ Lists in python:**

* **Python Lists are just like dynamically sized arrays, declared in other languages (vector in C++ and ArrayList 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. Tuples and String are other types of sequence data types.**
* **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 DataTypes like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.**
* **Lists in Python can be created by just placing the sequence inside the square brackets[].**
* **Unlike Sets, a list doesn’t need a built-in function for its creation of a list.**

**➢ Tuples in python:**

* **Tuples are used to store multiple items in a single variable.**
* **Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.**
* **A tuple is a collection which is ordered and unchangeable.**
* **Tuples are written with round brackets.**
* **Example:**

**thistuple = ("apple", "banana", "cherry")**

**print(thistuple)**

* **Tuple items are ordered, unchangeable, and allow duplicate values.**
* **Tuple items are indexed, the first item has index [0], the second item has index [1] etc.**
* **When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.**
* **Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.**
* **Since tuples are indexed, they can have items with the same value:**
* **Example:**
* **Tuples allow duplicate values:**

**thistuple = ("apple", "banana", "cherry", "apple", "cherry")**

**print(thistuple)**

**➢ Sets in python:**

* **A Set in Python programming is an unordered collection data type that is iterable, mutable and has no duplicate elements.**
* **Set are represented by { } (values enclosed in curly braces)**
* **The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set. This is based on a data structure known as a hash table. Since sets are unordered, we cannot access items using indexes as we do in lists.**
* **Sets are used to store multiple items in a single variable.**
* **Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.**
* **A set is a collection which is unordered, unchangeable\*, and unindexed.**
* **Set items are unchangeable, but you can remove items and add new items.**
* **Sets are written with curly brackets.**
* **Example:**

**thisset = {"apple", "banana", "cherry"}**

**print(thisset)**

* **Set items are unordered, unchangeable, and do not allow duplicate values.**
* **Unordered means that the items in a set do not have a defined order.**
* **Set items can appear in a different order every time you use them, and cannot be referred to by index or key. Set items are unchangeable, meaning that we cannot change the items after the set has been created.**
* **Once a set is created, you cannot change its items, but you can remove items and add new items.Sets cannot have two items with the same value.**

**➢ Dictionary in python:**

* **Dictionaries are used to store data values in key:value pairs.**
* **A dictionary is a collection which is ordered, changeable and do not allow duplicates.**
* **Dictionaries are written with curly brackets, and have keys and values:**
* **Example:**

**thisdict = {**

**"brand": "Ford",**

**"model": "Mustang",**

**"year": 1964**

**}**

**print(thisdict)**

* **Dictionary items are ordered, changeable, and do not allow duplicates.**
* **Dictionary items are presented in key:value pairs, and can be referred to by using the key**
* **name.**
* **Example:**
* **Print the "brand" value of the dictionary:**

**thisdict = {**

**"brand": "Ford",**

**"model": "Mustang",**

**"year": 1964**

**}print(thisdict["brand"])**

* **When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change.**
* **Unordered means that the items do not have a defined order, you cannot refer to an item by using an index.**
* **Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.**
* **Dictionaries cannot have two items with the same key.**

**➢ Functions in python:**

* **A function is a block of code which only runs when it is called.**
* **You can pass data, known as parameters, into a function.**
* **A function can return data as a result.**
* **In Python a function is defined using the def keyword:**
* **Example:**

**def my\_function():**

**print("Hello from a function")**

* **To call a function, use the function name followed by parenthesis:**
* **Example:**

**def my\_function():**

**print("Hello from a function")**

**my\_function()**

* **Information can be passed into functions as arguments.**
* **Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.**
* **The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:**
* **Example:**

**def my\_function(fname):**

**print(fname + " Refsnes")**

**my\_function("Emil")**

**my\_function("Tobias")**

**my\_function("Linus")**

* **The terms parameter and argument can be used for the same thing: information that are passed into a function.**
* **From a function's perspective:**
* **A parameter is the variable listed inside the parentheses in the function definition.**
* **An argument is the value that is sent to the function when it is called.**
* **By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.**
* **Example:**
* **This function expects 2 arguments, and gets 2 arguments:**

**def my\_function(fname, lname):**

**print(fname + " " + lname)**

**my\_function("Emil", "Refsnes")**

* **If you try to call the function with 1 or 3 arguments, you will get an error:**
* **Example:**
* **This function expects 2 arguments, but gets only 1:**

**def my\_function(fname, lname):**

**print(fname + " " + lname)**

**my\_function("Emil")**

* **If you do not know how many arguments that will be passed into your function, add**
* **a \* before the parameter name in the function definition.**
* **This way the function will receive a tuple of arguments, and can access the items**
* **accordingly:**
* **Example:**
* **If the number of arguments is unknown, add a \* before the parameter name:**

**def my\_function(\*kids):**

**print("The youngest child is " + kids[2])**

**my\_function("Emil", "Tobias", "Linus")**

* **You can also send arguments with the key = value syntax.**
* **This way the order of the arguments does not matter.**

**➢ Modules in python:**

* **Python Module is a file that contains built-in functions, classes, its and variables. There are many Python modules, each with its specific work.**
* **In this article, we will cover all about Python modules, such as How to create our own simple module, Import Python modules, From statements in Python, we can use the alias to rename the module, etc.**
* **A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code.**
* **Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.**
* **To create a Python module, write the desired code and save that in a file with .py extension. Let’s understand it better with an example:**
* **Example:**
* **Let’s create a simple calc.py in which we define two functions, one add and**
* **another subtract.**
* **# A simple module, calc.py**

**def add(x, y):**

**return (x+y)**

**def subtract(x, y):**

**return (x-y)**

* **Import module in Python:**
* **We can import the functions, and classes defined in a module to another module using**
* **the import statement in some other Python source file.**
* **When the interpreter encounters an import statement, it imports the module if the module is present in the search path.**
* **For example, to import the module calc.py, we need to put the following command at the top of the script.**
* **Syntax to Import Module in Python:**
* **import module**

**➢ Class & object in python:**

* **A class is a user-defined blueprint or prototype from which objects are created. Classes**
* **provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state.**
* **Class instances can also have methods**
* **(defined by their class) for modifying their state.**
* **To understand the need for creating a class and object in Python let’s consider an example, let’s say you wanted to track the number of dogs that may have different attributes like breed and age. If a list is used, the first element could be the dog’s breed while the second element could represent its age. Let’s suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it’s the exact need for classes.**
* **Syntax: Class Definition**
* **class ClassName:**
* **Some points on Python class:**
* **Classes are created by keyword class.**
* **Attributes are the variables that belong to a class.**
* **Attributes are always public and can be accessed using the dot (.) operator.**
* **Eg.: My class.Myattribute**
* **OBJECT:**
* **An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values. It’s not an idea anymore, it’s an actual dog, like a dog of breed pug who’s seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.**
* **An object consists of:**
* **State: It is represented by the attributes of an object. It also reflects the properties of an object.**
* **Behavior: It is represented by the methods of an object. It also reflects the response of an object to other objects.**
* **Identity: It gives a unique name to an object and enables one object to interact with other objects.**
* **Declaring Claas Objects (Also called instantiating a class):**
* **When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.**

**➢ File handling:**

* **File handling in Python is a powerful and versatile tool that can be used to perform a wide range of operations. However, it is important to carefully consider the advantages and disadvantages of file handling when writing Python programs, to ensure that the code is secure, reliable, and performs well.**
* **In this article we will explore Python File Handling, Advantages, Disadvantages and How open, write and append functions works in python file.**
* **Python supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, like other concepts of Python, this concept here is also easy and short. Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters, and they form a text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun.**
* **Let’s startwith the reading and writing files.**
* **Advantages of File Handling in Python:**
  + **Versatility: File handling in Python allows you to perform a wide range of operations, such as creating, reading, writing, appending, renaming, and deleting files.**
  + **Flexibility: File handling in Python is highly flexible, as it allows you to work with different file types (e.g. text files, binary files, CSV files, etc.), and to perform different operations on files (e.g. read, write, append, etc.).**
  + **User–friendly: Python provides a user-friendly interface for file handling, making it easy to create, read, and manipulate files.**
  + **Cross-platform: Python file-handling functions work across different platforms (e.g.**
  + **Windows, Mac, Linux), allowing for seamless integration and compatibility.**
  + **Disadvantages of File Handling in Python:**
  + **Error-prone: File handling operations in Python can be prone to errors, especially if the code is not carefully written or if there are issues with the file system (e.g. file permissions, file locks, etc.).**
  + **Security risks: File handling in Python can also pose security risks, especially if the program accepts user input that can be used to access or modify sensitive files on the system.**
  + **Complexity: File handling in Python can be complex, especially when working with more advanced file formats or operations. Careful attention must be paid to the code to ensure that files are handled properly and securely.**
  + **Performance: File handling operations in Python can be slower than other programming languages, especially when dealing with large files or performing complex operations.**

**➢ Exception handling:**

* **Error in Python can be of two types i.e. Syntax errors and Exceptions. Errors are problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which change the normal flow of the program.**
* **Different types of exceptions in python:**
* **In Python, there are several built-in Python exceptions that can be raised when an error occurs during the execution of a program. Here are some of the most common types of exceptions in Python:**
* **SyntaxError: This exception is raised when the interpreter encounters a syntax error in the code, such as a misspelled keyword, a missing colon, or an unbalanced parenthesis.**
* **TypeError: This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.**
* **NameError: This exception is raised when a variable or function name is not found in the current scope.**
* **IndexError: This exception is raised when an index is out of range for a list, tuple, or other sequence types.**
* **KeyError: This exception is raised when a key is not found in a dictionary.**
* **ValueError: This exception is raised when a function or method is called with an invalid**
* **argument or input, such as trying to convert a string to an integer when the string does not represent a valid integer.**
* **IOError: This exception is raised when an I/O operation, such as reading or writing a file,fails due to an input/output error.**
* **ZeroDivisionError: This exception is raised when an attempt is made to divide a number by Zero**

**CODE:**

**Registration/templates/registration/login.html**

{% extends "registration/RegisterBase.html" %}  
{% block title %} Login {% endblock %}  
{% load crispy\_forms\_tags %}  
  
{% block content %}  
 <div class="sign-up form" style="max-width: 600px">  
 <h1 class="mt-2">User Login</h1>  
 <hr class="mt-0 mb-4">  
 <form method="post" class="form-group">  
 {% csrf\_token %}  
 {{form|crispy}}  
 <p>Don't have an account?<a href="/signup"> Create one.</a></p>  
 <button type="submit" class="btn btn-success">Login</button>  
  
 </form>  
 </div>  
{% endblock %}

**Registration/templates/registration/signup.html**

{% extends "registration/RegisterBase.html" %}  
{% block title %} Sign Up {% endblock %}  
  
{% load crispy\_forms\_tags %}  
 {% for msg in message %}  
 <script>  
 window.alert(msg)  
 </script>  
 {% endfor %}  
{% block content %}  
  
 <div class="sign-up form" style="max-width: 600px">  
 <h1 class="mt-2">{{heading}}</h1>  
 <hr class="mt-0 mb-4">  
  
 <form method="post" class="form-group" style="max-width: 600px;">  
 {% csrf\_token %}  
 {{form|crispy}}  
 <p>Already have an account?<a href="/login">Login here.</a></p>  
 <button type="submit" class="btn btn-success" >Sign Up</button>  
  
 </form>  
 </div>  
{% endblock %}

**Registration/forms.py**

from django import forms  
from django.contrib.auth.forms import UserCreationForm  
from django.contrib.auth.models import User  
from chat.models import UserProfile  
  
  
class SignUpForm(forms.Form):  
  
 username = forms.CharField(min\_length=5, max\_length=20)  
 name = forms.CharField(max\_length=25, label="Name")  
 email = forms.EmailField(label="Email")  
 password1 = forms.CharField(label='Password', widget=forms.PasswordInput)  
 password2 = forms.CharField(label="Confirm Password", widget=forms.PasswordInput)  
  
 def validate\_username(self):  
 username = self.cleaned\_data['username']  
 r = User.objects.filter(username=username)  
 if r.count():  
 return None  
 return username  
  
 def validate\_password(self):  
 password1 = self.cleaned\_data['password1']  
 password2 = self.cleaned\_data['password2']  
 if not password1 or not password2:  
 return None  
 elif password2 != password1:  
 return None  
 return password1  
  
 def validate\_email(self):  
 email = self.cleaned\_data['email']  
 r = UserProfile.objects.filter(email=email)  
 if r.count():  
 return None  
 return email  
  
 def save(self, commit=True):  
 user = User.objects.create\_user(  
 self.cleaned\_data['username'],  
 self.cleaned\_data['email'],  
 self.cleaned\_data['password1']  
 )  
 return user

**Registration/views.py**

from django.shortcuts import render, redirect  
from .forms import SignUpForm  
from django.contrib.auth import login, authenticate  
from chat.models import UserProfile  
  
  
def SignUp(request):  
 *"""  
 Sign up view* ***:param*** *request:* ***:return****:  
 """* message = []  
 if request.method == "POST":  
 form = SignUpForm(request.POST)  
 if form.is\_valid():  
 name = form.cleaned\_data.get('name')  
 email = form.validate\_email()  
 username = form.validate\_username()  
 password = form.validate\_password()  
 if not email:  
 message.append("Email already registered!")  
 elif not password:  
 message.append("Passwords don't match!")  
 elif not username:  
 message.append("Username already registered!")  
 else:  
 print("SUCCESS!!!!")  
 form.save()  
 user = authenticate(username=username, password=password)  
 login(request, user)  
 profile = UserProfile(email=email, name=name, username=username)  
 profile.save()  
 return redirect("/")  
 else:  
 form = SignUpForm()  
 return render(request, "registration/signup.html", {"form": form, "heading": "Sign Up", "message": message})

chat/views.py

from django.shortcuts import render, HttpResponse, redirect  
from .models import UserProfile, Friends, Messages  
from django.views.decorators.csrf import csrf\_exempt  
from django.http.response import JsonResponse  
from rest\_framework.parsers import JSONParser  
from chat.serializers import MessageSerializer  
  
  
def getFriendsList(id):  
 *"""  
 Get the list of friends of the user* ***:param****: user id* ***:return****: list of friends  
 """* try:  
 user = UserProfile.objects.get(id=id)  
 ids = list(user.friends\_set.all())  
 friends = []  
 for id in ids:  
 num = str(id)  
 fr = UserProfile.objects.get(id=int(num))  
 friends.append(fr)  
 return friends  
 except:  
 return []  
  
  
def getUserId(username):  
 *"""  
 Get the user id by the username* ***:param*** *username:* ***:return****: int  
 """* use = UserProfile.objects.get(username=username)  
 id = use.id  
 return id  
  
  
def index(request):  
 *"""  
 Return the home page* ***:param*** *request:* ***:return****:  
 """* if not request.user.is\_authenticated:  
 print("Not Logged In!")  
 return render(request, "chat/index.html", {})  
 else:  
 username = request.user.username  
 id = getUserId(username)  
 friends = getFriendsList(id)  
 return render(request, "chat/Base.html", {'friends': friends})  
  
  
def search(request):  
 *"""  
 Search users page* ***:param*** *request:* ***:return****:  
 """* users = list(UserProfile.objects.all())  
 for user in users:  
 if user.username == request.user.username:  
 users.remove(user)  
 break  
  
 if request.method == "POST":  
 print("SEARCHING!!")  
 query = request.POST.get("search")  
 user\_ls = []  
 for user in users:  
 if query in user.name or query in user.username:  
 user\_ls.append(user)  
 return render(request, "chat/search.html", {'users': user\_ls, })  
  
 try:  
 users = users[:10]  
 except:  
 users = users[:]  
 id = getUserId(request.user.username)  
 friends = getFriendsList(id)  
 return render(request, "chat/search.html", {'users': users, 'friends': friends})  
  
  
def addFriend(request, name):  
 *"""  
 Add a user to the friend's list  
 :param request:  
 :param name:  
 :return:  
 """* username = request.user.username  
 id = getUserId(username)  
 friend = UserProfile.objects.get(username=name)  
 curr\_user = UserProfile.objects.get(id=id)  
 print(curr\_user.name)  
 ls = curr\_user.friends\_set.all()  
 flag = 0  
 for username in ls:  
 if username.friend == friend.id:  
 flag = 1  
 break  
 if flag == 0:  
 print("Friend Added!!")  
 curr\_user.friends\_set.create(friend=friend.id)  
 friend.friends\_set.create(friend=id)  
 return redirect("/search")  
  
  
def chat(request, username):  
 *"""  
 Get the chat between two users.  
 :param request:  
 :param username:  
 :return:  
 """* friend = UserProfile.objects.get(username=username)  
 id = getUserId(request.user.username)  
 curr\_user = UserProfile.objects.get(id=id)  
 messages = Messages.objects.filter(sender\_name=id, receiver\_name=friend.id) | Messages.objects.filter(sender\_name=friend.id, receiver\_name=id)  
  
 if request.method == "GET":  
 friends = getFriendsList(id)  
 return render(request, "chat/messages.html",  
 {'messages': messages,  
 'friends': friends,  
 'curr\_user': curr\_user, 'friend': friend})  
  
  
@csrf\_exempt  
def message\_list(request, sender=None, receiver=None):  
 if request.method == 'GET':  
 messages = Messages.objects.filter(sender\_name=sender, receiver\_name=receiver, seen=False)  
 serializer = MessageSerializer(messages, many=True, context={'request': request})  
 for message in messages:  
 message.seen = True  
 message.save()  
 return JsonResponse(serializer.data, safe=False)  
  
 elif request.method == "POST":  
 data = JSONParser().parse(request)  
 serializer = MessageSerializer(data=data)  
 if serializer.is\_valid():  
 serializer.save()  
 return JsonResponse(serializer.data, status=201)  
 return JsonResponse(serializer.errors, status=400)

**chat/models.py:**

from django.db import models  
  
  
class UserProfile(models.Model):  
  
 name = models.CharField(max\_length=25)  
 email = models.EmailField(unique=True)  
 username = models.CharField(max\_length=20, unique=True)  
  
 def \_\_str\_\_(self):  
 return f"{self.name}"  
  
  
class Messages(models.Model):  
  
 description = models.TextField()  
 sender\_name = models.ForeignKey(UserProfile, on\_delete=models.CASCADE, related\_name='sender')  
 receiver\_name = models.ForeignKey(UserProfile, on\_delete=models.CASCADE, related\_name='receiver')  
 time = models.TimeField(auto\_now\_add=True)  
 seen = models.BooleanField(default=False)  
 timestamp = models.DateTimeField(auto\_now\_add=True)  
  
 def \_\_str\_\_(self):  
 return f"To: {self.receiver\_name} From: {self.sender\_name}"  
  
 class Meta:  
 ordering = ('timestamp',)  
  
  
class Friends(models.Model):  
  
 user = models.ForeignKey(UserProfile, on\_delete=models.CASCADE)  
 friend = models.IntegerField()  
  
 def \_\_str\_\_(self):  
 return f"{self.friend}"

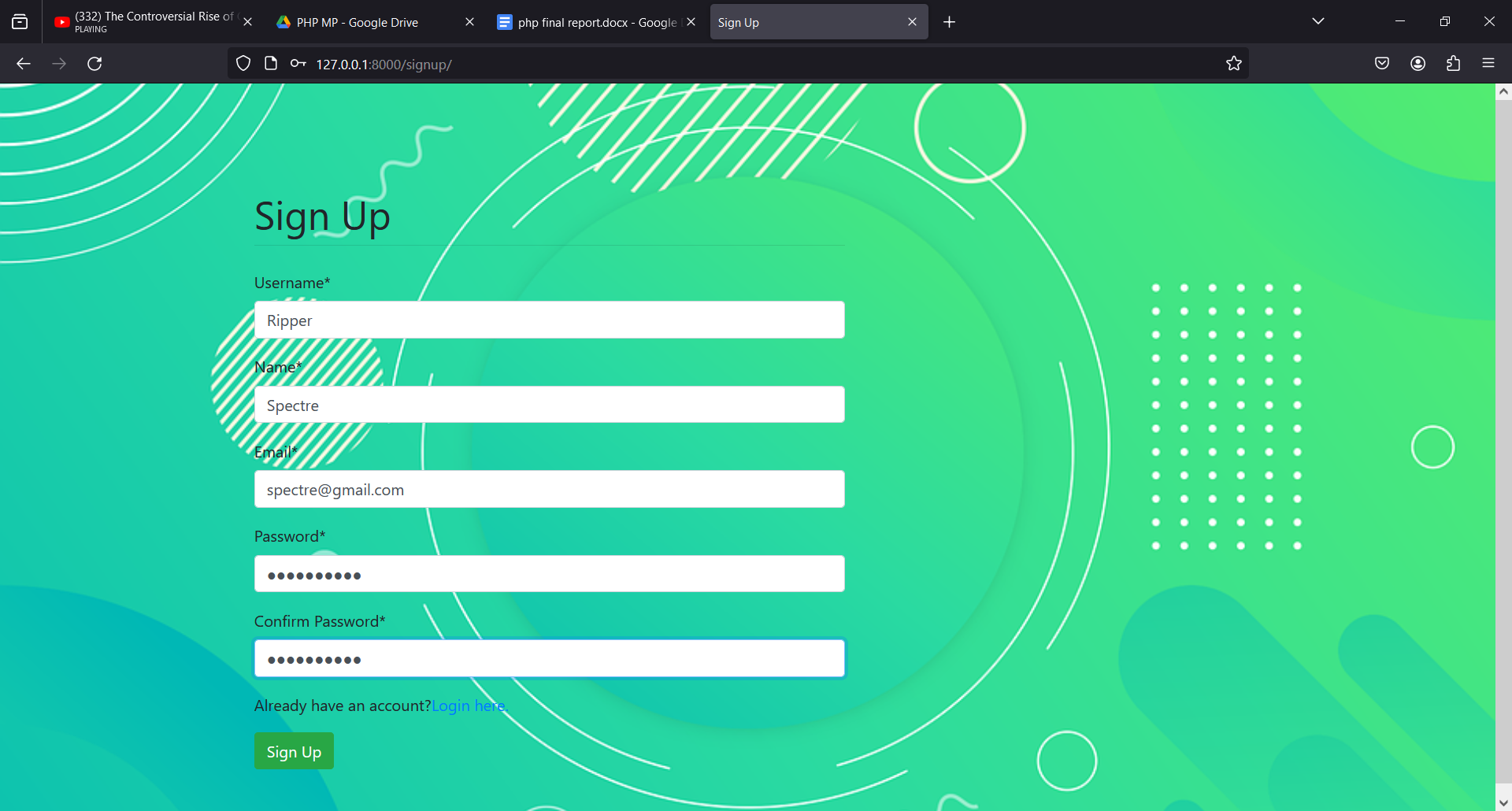
**static/js.chat.js**

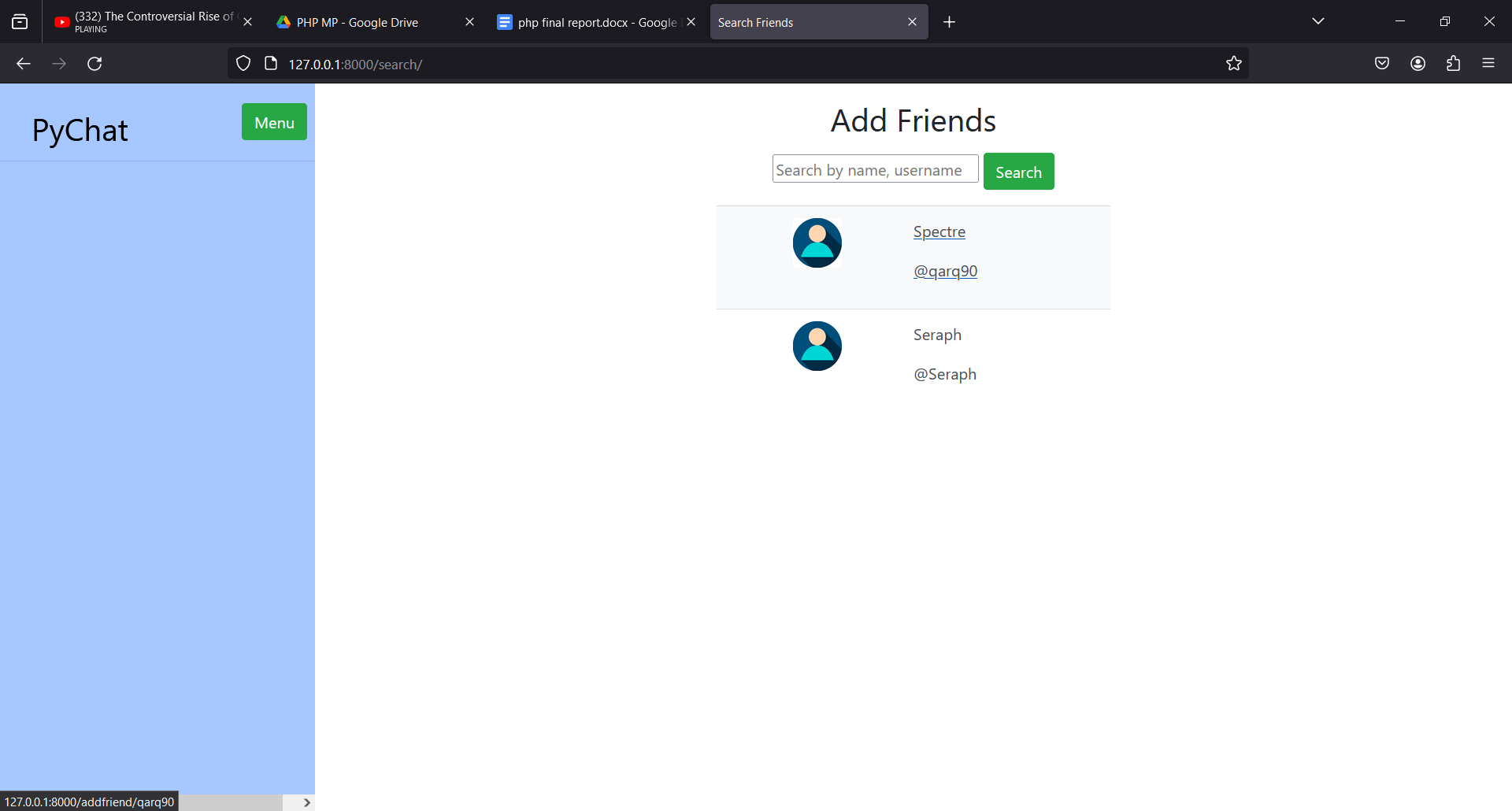
const ***text\_box*** = '<div class="container darker">' +  
 '<img src="{% static 'images/user\_image.jpg' %}" alt="Avatar" class="right" style="width:100%;">'  
 + '<p>{message}</p>' +  
 '<span class="time-right">{time}</span>'  
 + '</div>';  
  
function send(sender, receiver, message, time){  
 ***console***.log("YES! WORKING")  
 $.post('api/messages', '{"sender:" "' + sender + '", "receiver": "' + receiver + '", "message": "' +  
 message + '"}', function(data){  
 var field = ***text\_box***.replace('{message}', message);  
 field.replace('{time}', time)  
 $('#board').append(field);  
 })  
}  
  
function receive(){  
 $.get('api/messages/' + sender\_id + '/' + receiver\_id, function(data){  
 ***console***.log(data);  
 if(data.length() !== 0){  
 for(var i = 0; i < data.length; i++){  
 var field = ***text\_box***.replace('{message}', message);  
 $('#board').append(field);  
 }  
 }  
 })  
}

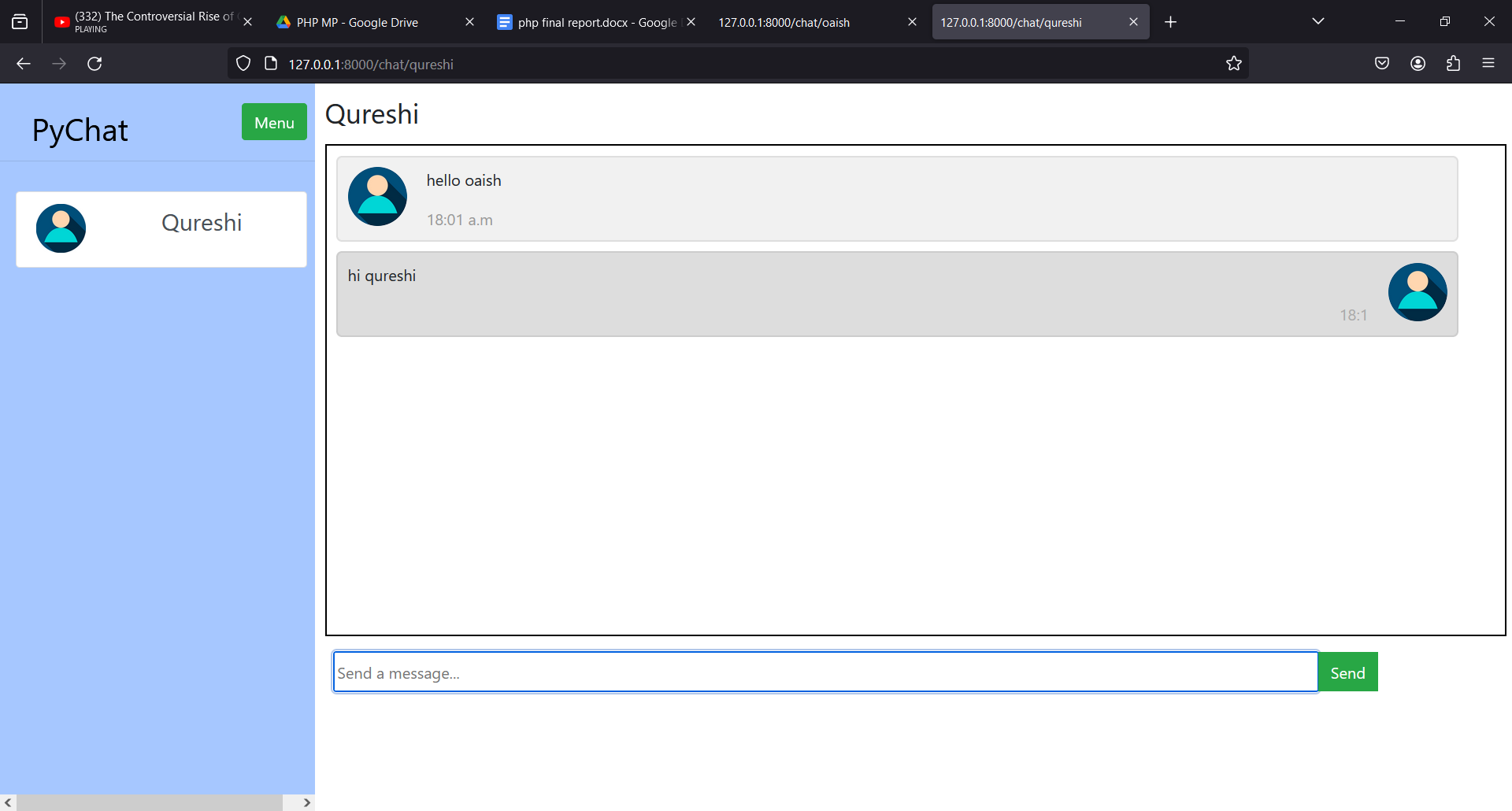
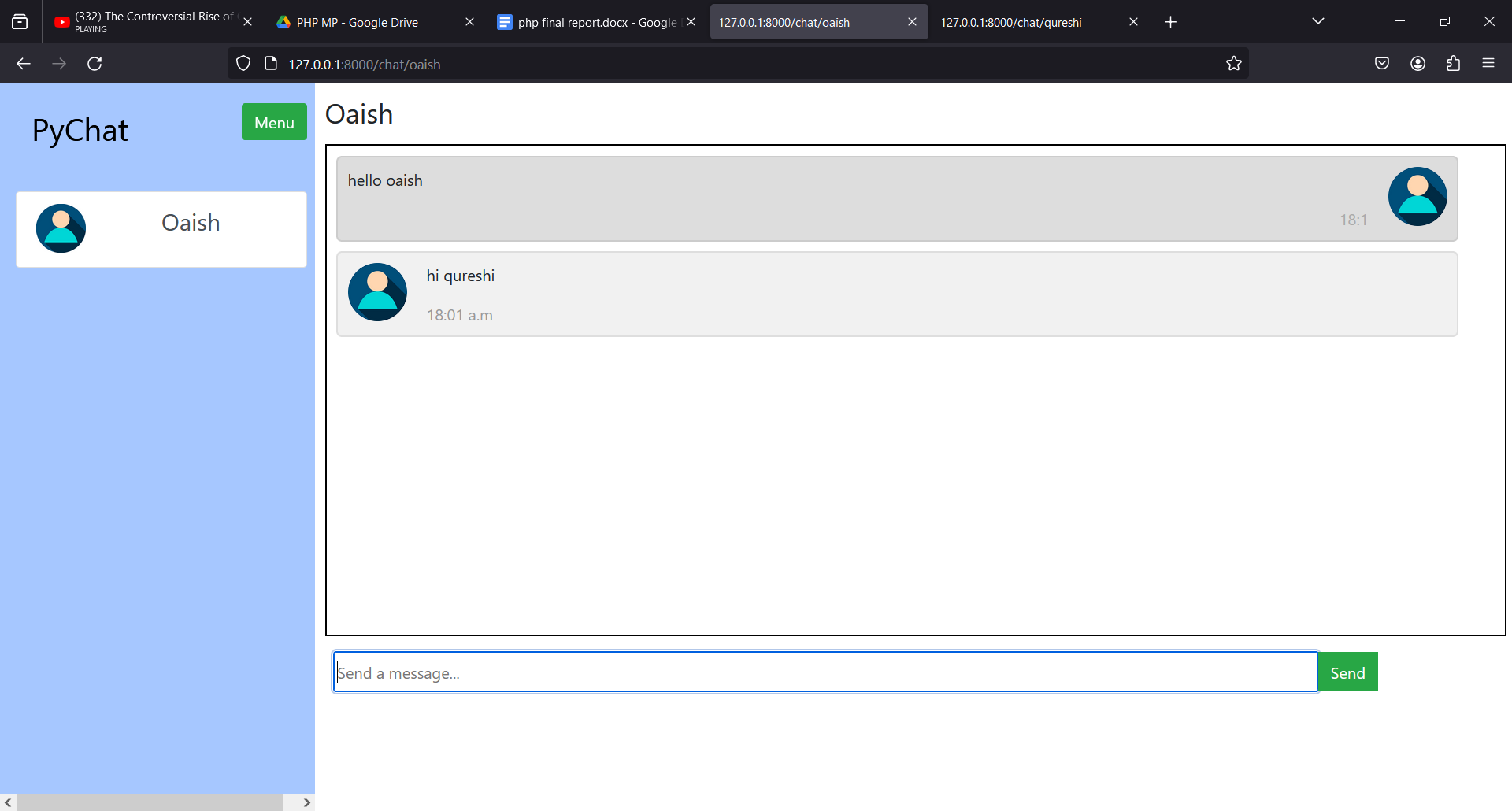
**rest\_framework/docs/js/api.js**

var ***responseDisplay*** = 'data'  
var ***coreapi*** = ***window***.***coreapi***var ***schema*** = ***window***.***schema***function normalizeKeys(arr) {  
 var \_normarr = [];  
 for (var i = 0; i < arr.length; i++) {  
 \_normarr = \_normarr.concat(arr[i].split(' > '));  
 }  
 return \_normarr;  
}  
  
function normalizeHTTPHeader(str) {  
 *// Capitalize HTTP headers for display.* return (str.charAt(0).toUpperCase() + str.substring(1))  
 .replace(/-(.)/g, function ($1) {  
 return $1.toUpperCase()  
 })  
 .replace(/(Www)/g, function ($1) {  
 return 'WWW'  
 })  
 .replace(/(Xss)/g, function ($1) {  
 return 'XSS'  
 })  
 .replace(/(Md5)/g, function ($1) {  
 return 'MD5'  
 })  
}  
  
function formEntries(form) {  
 *// Polyfill for new FormData(form).entries()* var formData = new FormData(form)  
 if (formData.entries !== undefined) {  
 return ***Array***.from(formData.entries())  
 }  
  
 var entries = []  
  
 for (var i = 0; i < form.elements.length; i++) {  
 var element = form.elements[i]  
  
 if (!element.name) {  
 continue  
 }  
  
 if (element.type === 'file') {  
 for (var j = 0; j < element.files.length; j++) {  
 entries.push([element.name, element.files[j]])  
 }  
 } else if (element.type === 'select-multiple' || element.type === 'select-one') {  
 for (var j = 0; j < element.selectedOptions.length; j++) {  
 entries.push([element.name, element.selectedOptions[j].value])  
 }  
 } else if (element.type === 'checkbox') {  
 if (element.checked) {  
 entries.push([element.name, element.value])  
 }  
 } else {  
 entries.push([element.name, element.value])  
 }  
 }  
  
 return entries  
}  
  
$(function () {  
 var $selectedAuthentication = $('#selected-authentication')  
 var $authControl = $('#auth-control')  
 var $authTokenModal = $('#auth\_token\_modal')  
 var $authBasicModal = $('#auth\_basic\_modal')  
 var $authSessionModal = $('#auth\_session\_modal')  
  
 *// Language Control* $('#language-control li').click(function (event) {  
 event.preventDefault()  
 var $languageMenuItem = $(this).find('a')  
 var $languageControls = $(this).closest('ul').find('li')  
 var $languageControlLinks = $languageControls.find('a')  
 var language = $languageMenuItem.data('language')  
  
 $languageControlLinks.not('[data-language="' + language + '"]').parent().removeClass('active')  
 $languageControlLinks.filter('[data-language="' + language + '"]').parent().addClass('active')  
  
 $('#selected-language').text(language)  
  
 var $codeBlocks = $('pre.highlight')  
 $codeBlocks.not('[data-language="' + language + '"]').addClass('hide')  
 $codeBlocks.filter('[data-language="' + language + '"]').removeClass('hide')  
 })  
  
 *// API Explorer* $('form.api-interaction').submit(function (event) {  
 event.preventDefault()  
  
 var $form = $(this).closest('form')  
 var $requestMethod = $form.find('.request-method')  
 var $requestUrl = $form.find('.request-url')  
 var $toggleView = $form.closest('.modal-content').find('.toggle-view')  
 var $responseStatusCode = $form.find('.response-status-code')  
 var $meta = $form.find('.meta')  
 var $responseRawResponse = $form.find('.response-raw-response')  
 var $requestAwaiting = $form.find('.request-awaiting')  
 var $responseRaw = $form.find('.response-raw')  
 var $responseData = $form.find('.response-data')  
 var key = normalizeKeys($form.data('key'))  
 var params = {}  
 var entries = formEntries($form.get()[0])  
  
 for (var i = 0; i < entries.length; i++) {  
 var entry = entries[i]  
 var paramKey = entry[0]  
 var paramValue = entry[1]  
 var $elem = $form.find('[name="' + paramKey + '"]')  
 var dataType = $elem.data('type') || 'string'  
  
 if (dataType === 'integer' && paramValue) {  
 var value = parseInt(paramValue)  
 if (!isNaN(value)) {  
 params[paramKey] = value  
 }  
 } else if (dataType === 'number' && paramValue) {  
 var value = parseFloat(paramValue)  
 if (!isNaN(value)) {  
 params[paramKey] = value  
 }  
 } else if (dataType === 'boolean' && paramValue) {  
 var value = {  
 'true': true,  
 'false': false  
 }[paramValue.toLowerCase()]  
 if (value !== undefined) {  
 params[paramKey] = value  
 }  
 } else if (dataType === 'array' && paramValue) {  
 try {  
 params[paramKey] = ***JSON***.parse(paramValue)  
 } catch (err) {  
 *// Ignore malformed JSON* }  
 } else if (dataType === 'object' && paramValue) {  
 try {  
 params[paramKey] = ***JSON***.parse(paramValue)  
 } catch (err) {  
 *// Ignore malformed JSON* }  
 } else if (dataType === 'string' && paramValue) {  
 params[paramKey] = paramValue  
 }  
 }  
  
 $form.find(':checkbox').each(function (index) {  
 *// Handle unselected checkboxes* var name = $(this).attr('name')  
 if (!params.hasOwnProperty(name)) {  
 params[name] = false  
 }  
 })  
  
 function requestCallback(request) {  
 *// Fill in the "GET /foo/" display.* var parser = ***document***.createElement('a')  
 parser.href = request.url  
 var method = request.options.method  
 var path = parser.pathname + parser.hash + parser.search  
  
 $requestMethod.text(method)  
 $requestUrl.text(path)  
 }  
  
 function responseCallback(response, responseText) {  
 *// Display the 'Data'/'Raw' control.* $toggleView.removeClass('hide')  
  
 *// Fill in the "200 OK" display.* $responseStatusCode.removeClass('label-success').removeClass('label-danger')  
 if (response.ok) {  
 $responseStatusCode.addClass('label-success')  
 } else {  
 $responseStatusCode.addClass('label-danger')  
 }  
 $responseStatusCode.text(response.status)  
 $meta.removeClass('hide')  
  
 *// Fill in the Raw HTTP response display.* var panelText = 'HTTP/1.1 ' + response.status + ' ' + response.statusText + '\n'  
 response.headers.forEach(function (header, key) {  
 panelText += normalizeHTTPHeader(key) + ': ' + header + '\n'  
 })  
 if (responseText) {  
 panelText += '\n' + responseText  
 }  
 $responseRawResponse.text(panelText)  
 }  
  
 *// Instantiate a client to make the outgoing request.* var options = {  
 requestCallback: requestCallback,  
 responseCallback: responseCallback  
 }  
  
 *// Setup authentication options.* if (***window***.auth && ***window***.auth.type === 'token') {  
 *// Header authentication* options.auth = new ***coreapi***.auth.TokenAuthentication({  
 scheme: ***window***.auth.scheme,  
 token: ***window***.auth.token  
 })  
 } else if (***window***.auth && ***window***.auth.type === 'basic') {  
 *// Basic authentication* options.auth = new ***coreapi***.auth.BasicAuthentication({  
 username: ***window***.auth.username,  
 password: ***window***.auth.password  
 })  
 } else if (***window***.auth && ***window***.auth.type === 'session') {  
 *// Session authentication* options.auth = new ***coreapi***.auth.SessionAuthentication({  
 csrfCookieName: 'csrftoken',  
 csrfHeaderName: 'X-CSRFToken'  
 })  
 }  
  
 var client = new ***coreapi***.Client(options)  
 client.action(***schema***, key, params).then(function (data) {  
 var response = ***JSON***.stringify(data, null, 2)  
 $requestAwaiting.addClass('hide')  
 $responseRaw.addClass('hide')  
 $responseData.addClass('hide').text('').jsonView(response)  
  
 if (***responseDisplay*** === 'data') {  
 $responseData.removeClass('hide')  
 } else {  
 $responseRaw.removeClass('hide')  
 }  
 }).catch(function (error) {  
 var response = ***JSON***.stringify(error.content, null, 2)  
 $requestAwaiting.addClass('hide')  
 $responseRaw.addClass('hide')  
 $responseData.addClass('hide').text('').jsonView(response)  
  
 if (***responseDisplay*** === 'data') {  
 $responseData.removeClass('hide')  
 } else {  
 $responseRaw.removeClass('hide')  
 }  
 })  
 })  
  
 *// 'Data'/'Raw' control* $('.toggle-view button').click(function () {  
 var $modalContent = $(this).closest('.modal-content')  
 var $modalResponseRaw = $modalContent.find('.response-raw')  
 var $modalResponseData = $modalContent.find('.response-data')  
  
 ***responseDisplay*** = $(this).data('display-toggle')  
  
 $(this).removeClass('btn-default').addClass('btn-info').siblings().removeClass('btn-info')  
  
 if (***responseDisplay*** === 'raw') {  
 $modalResponseRaw.removeClass('hide')  
 $modalResponseData.addClass('hide')  
 } else {  
 $modalResponseData.removeClass('hide')  
 $modalResponseRaw.addClass('hide')  
 }  
 })  
  
 *// Authentication: none* $authControl.find("[data-auth='none']").click(function (event) {  
 event.preventDefault()  
 ***window***.auth = null  
 $selectedAuthentication.text('none')  
 $authControl.find("[data-auth]").closest('li').removeClass('active')  
 $authControl.find("[data-auth='none']").closest('li').addClass('active')  
 })  
  
 *// Authentication: token* $('form.authentication-token-form').submit(function (event) {  
 event.preventDefault()  
 var $form = $(this).closest('form')  
 var scheme = $form.find('input#scheme').val()  
 var token = $form.find('input#token').val()  
 ***window***.auth = {  
 'type': 'token',  
 'scheme': scheme,  
 'token': token  
 }  
 $selectedAuthentication.text('token')  
 $authControl.find("[data-auth]").closest('li').removeClass('active')  
 $authControl.find("[data-auth='token']").closest('li').addClass('active')  
 $authTokenModal.modal('hide')  
 })  
  
 *// Authentication: basic* $('form.authentication-basic-form').submit(function (event) {  
 event.preventDefault()  
 var $form = $(this).closest('form')  
 var username = $form.find('input#username').val()  
 var password = $form.find('input#password').val()  
 ***window***.auth = {  
 'type': 'basic',  
 'username': username,  
 'password': password  
 }  
 $selectedAuthentication.text('basic')  
 $authControl.find("[data-auth]").closest('li').removeClass('active')  
 $authControl.find("[data-auth='basic']").closest('li').addClass('active')  
 $authBasicModal.modal('hide')  
 })  
  
 *// Authentication: session* $('form.authentication-session-form').submit(function (event) {  
 event.preventDefault()  
 ***window***.auth = {  
 'type': 'session'  
 }  
 $selectedAuthentication.text('session')  
 $authControl.find("[data-auth]").closest('li').removeClass('active')  
 $authControl.find("[data-auth='session']").closest('li').addClass('active')  
 $authSessionModal.modal('hide')  
 })  
})

**Output:**





**8.0 Skills Developed** / **Learning outcome of this Micro-Project**

The following skills are developed:

1) **Identifying:** Identifying the problem and cause of problem in the area related and prepare project proposals before starting the project.

2) **Designing:** Designing of micro project with minimum required resources (low cost).

3) **Teamwork:** Learn to work in a team and boost individual confidence.

4) **Time Management:** Timely completion of micro project as scheduled.

5) **Problem-solving:** Develop good problem-solving skills.

6) **Technical Writing:** Preparing a report of the proposed plan and final report.

7) **Confidence:** Confidently, answer the questions asked about the project.

8) **Acknowledgement:** Acknowledge the help rendered by others in the success of the project.

**9.0 Applications of this Micro-Project**

* Educational Tool
* Teaching Resource
* Personal Portfolio
* Intranet Communication

**(To be evaluated by the concerned teacher)**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Annexure IV**

**Micro Project Evaluation Sheet**

**Name of Student: Abdurrahman Qureshi Enrollment No.: 2100020112**

**Name of Programme: Computer Engineering Semester: SIXTH**

**Course Title: Programming with python (PWP) Code: 22616**

**Title of the Micro-Project: Chat Application**

**Course Outcomes Achieved: -**

* Develop python program to demonstrate use of Operators
* Perform operations on data structures in Python.
* Develop functions for given problem.
* Design classes for given problem.
* Handle exceptions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr No.** | **Characteristics to be assessed** | **Poor**  **(Marks 1 - 3)** | **Average**  **(Marks 4 - 5)** | **Good**  **(Marks 6 - 8)** | **Excellent**  **(Marks 9- 10)** | **Sub Total** |
| (A) Process and Product Assessment (Convert above total marks out of 6 Marks) | | | | | | |
| 1 | Relevance to the course |  |  |  |  |  |
| 2 | Literature Review/information collection |  |  |  |  |
| 3 | Completion of the Target as per project proposal |  |  |  |  |
| 4 | Analysis of Data and representation |  |  |  |  |
| 5 | Quality of Prototype/Model |  |  |  |  |
| 6 | Report Preparation |  |  |  |  |
| (B) Individual Presentation/ Viva (Convert above total marks out of 4 Marks) | | | | | | |
| 7 | Presentation |  |  |  |  |  |
| 8 | Viva |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **(A)**  **Process and Product Assessment**  **(6 marks)** | **(B)**  **Individual Presentation & viva**  **(4 marks)** | **Total Marks**  **10** |
|  |  |  |

**Comments/Suggestions about teamwork/leadership/inter-personal communication (if any)**

**............................................................................................................................................................................**

**............................................................................................................................................................................**

**............................................................................................................................................................................**

**Name and designation of the Teacher Prof. Zaibunnisa Malik Ma’am**

**Dated Signature .................................................................**

**Micro Project Evaluation Sheet**

**Name of Student: Qazi Mohd Oaish Enrollment No.: 2100020108**

**Name of Programme: Computer Engineering Semester: SIXTH**

**Course Title: Programming with python (PWP) Code: 22616**

**Title of the Micro-Project: Chat Application**

**Course Outcomes Achieved: -**

* Develop python program to demonstrate use of Operators
* Perform operations on data structures in Python.
* Develop functions for given problem.
* Design classes for given problem.
* Handle exceptions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr No.** | **Characteristics to be assessed** | **Poor**  **(Marks 1 - 3)** | **Average**  **(Marks 4 - 5)** | **Good**  **(Marks 6 - 8)** | **Excellent**  **(Marks 9- 10)** | **Sub Total** |
| (A) Process and Product Assessment (Convert above total marks out of 6 Marks) | | | | | | |
| 1 | Relevance to the course |  |  |  |  |  |
| 2 | Literature Review/information collection |  |  |  |  |
| 3 | Completion of the Target as per project proposal |  |  |  |  |
| 4 | Analysis of Data and representation |  |  |  |  |
| 5 | Quality of Prototype/Model |  |  |  |  |
| 6 | Report Preparation |  |  |  |  |
| (B) Individual Presentation/ Viva (Convert above total marks out of 4 Marks) | | | | | | |
| 7 | Presentation |  |  |  |  |  |
| 8 | Viva |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **(A)**  **Process and Product Assessment**  **(6 marks)** | **(B)**  **Individual Presentation & viva**  **(4 marks)** | **Total Marks**  **10** |
|  |  |  |

**Comments/Suggestions about teamwork/leadership/inter-personal communication (if any)**

**............................................................................................................................................................................**

**............................................................................................................................................................................**

**............................................................................................................................................................................**

**Name and designation of the Teacher Prof. Zaibunnisa Malik Ma’am**

**Dated Signature .................................................................**

**Micro Project Evaluation Sheet**

**Name of Student: Shaikh Mohammed Hussain Enrollment No.: 2200020625**

**Name of Programme: Computer Engineering Semester: SIXTH**

**Course Title: Programming with python (PWP) Code: 22616**

**Title of the Micro-Project: Chat Application**

**Course Outcomes Achieved: -**

* Develop python program to demonstrate use of Operators
* Perform operations on data structures in Python.
* Develop functions for given problem.
* Design classes for given problem.
* Handle exceptions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr No.** | **Characteristics to be assessed** | **Poor**  **(Marks 1 - 3)** | **Average**  **(Marks 4 - 5)** | **Good**  **(Marks 6 - 8)** | **Excellent**  **(Marks 9- 10)** | **Sub Total** |
| (A) Process and Product Assessment (Convert above total marks out of 6 Marks) | | | | | | |
| 1 | Relevance to the course |  |  |  |  |  |
| 2 | Literature Review/information collection |  |  |  |  |
| 3 | Completion of the Target as per project proposal |  |  |  |  |
| 4 | Analysis of Data and representation |  |  |  |  |
| 5 | Quality of Prototype/Model |  |  |  |  |
| 6 | Report Preparation |  |  |  |  |
| (B) Individual Presentation/ Viva (Convert above total marks out of 4 Marks) | | | | | | |
| 7 | Presentation |  |  |  |  |  |
| 8 | Viva |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **(A)**  **Process and Product Assessment**  **(6 marks)** | **(B)**  **Individual Presentation & viva**  **(4 marks)** | **Total Marks**  **10** |
|  |  |  |

**Comments/Suggestions about teamwork/leadership/inter-personal communication (if any)**

**............................................................................................................................................................................**

**............................................................................................................................................................................**

**............................................................................................................................................................................**

**Name and designation of the Teacher Prof. Zaibunnisa Malik Ma’am**

**Dated Signature .................................................................**