**Chatting Web App**

Bachelor of Technology (Computer Science & Engineering)

(Session: 2018-2022)



**Submitted to: Submitted by:**

Mr. Amandeep Parag Negi

CSE Department 251802038

UIET, KUK CSE-A 7th Sem.

**University Institute of Engineering and Technology**

**Kurukshetra University, Kurukshetra – 136119**

## CONTENTS

* Acknowledgements
* Certificate
* Introduction
* Prerequisites
* Working
* Code
* Output

## ACKNOWLEDGEMENT

This formal piece of acknowledgement is an attempt to express the feeling of Gratitude towards people who helped me in successfully completing my project.

I would like to express my special thanks of gratitude to my mentors at internshala trainings who gave me the golden opportunity to do this wonderful project on the topic “Chatting Web App” which also helped me in doing a lot of research and I came to know about so many new things. I am really thankful to do this project work.

Secondly, I would like to thank my parents and my friends who helped me a lot in finishing this project.

Parag Negi

251802038

# CERTIFICATE



# Introduction

* PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

* HTML

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript).

"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li> and many others.

An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the <title> tag can be written as <Title>, <TITLE>, or in any other way.

* JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

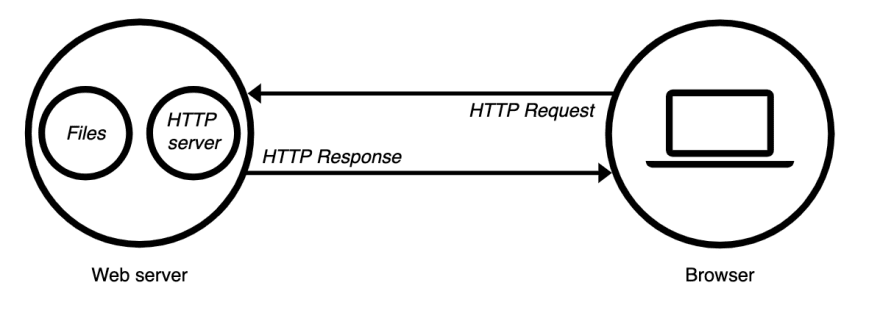
# Prerequisites

## Client-Server Architecture

[![Photo by [İsmail Enes Ayhan](https://unsplash.com/@ismailenesayhan?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)](data:image/jpeg;base64,)](https://res.cloudinary.com/practicaldev/image/fetch/s--vUoX0dxT--/c_limit,f_auto,fl_progressive,q_auto,w_880/https:/cdn-images-1.medium.com/max/800/0*7BucOLCRTOJ_m66S)

The client-server architecture is a basic computing model where the client requests services from the server, while the server processes these requests and provides services to the clients.

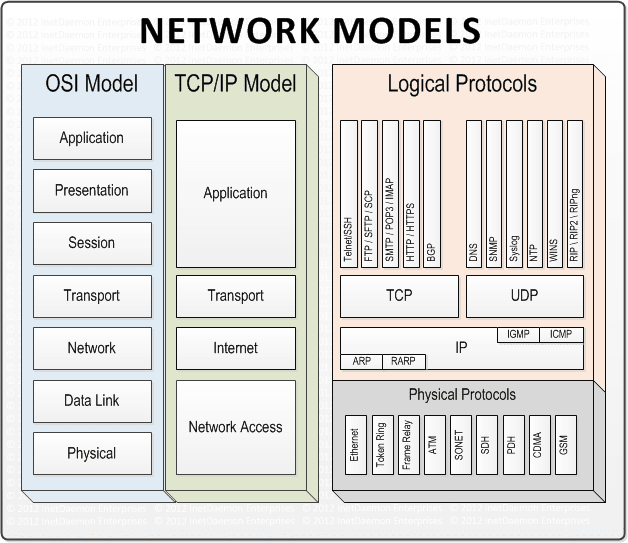
In fact, your web browser is a client requesting web services from Medium’s web server through the HyperText Transfer Protocol (HTTP) right now! Medium’s web server processes your request and returns HTML documents, CSS stylesheets, JavaScript files and images so that your web browser can display the website the way it is.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--DyXC-dDA--/c_limit,f_auto,fl_progressive,q_auto,w_880/https:/cdn-images-1.medium.com/max/2392/1*DixueByHI55JwHRBo9DPYA.png)

Furthermore, a server can serve multiple clients! In our application, we want many clients to be talking in real-time. The client software will send a message to the chatroom server, and the chatroom server will broadcast our message to all other connected clients.

## Protocols, Protocols, Protocols

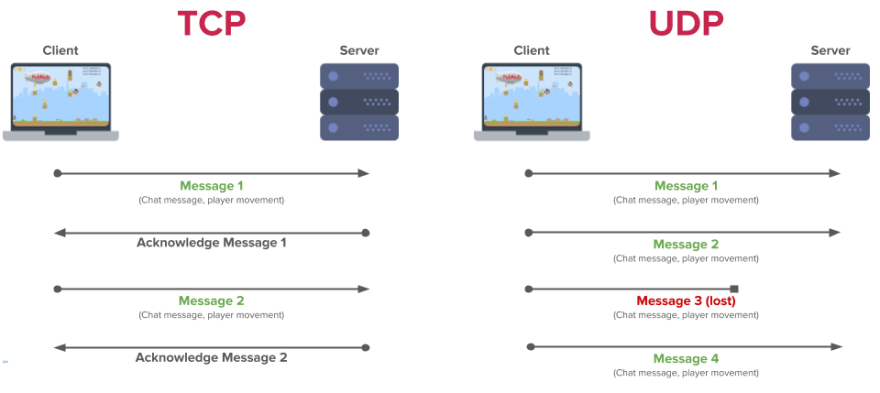
Communications over networks use what we call a protocol stack — building higher-level, more sophisticated conversations on top of simpler, more rudimentary conversations. Network layers can be represented by the OSI model and the TCP/IP model. Each network layer corresponds to a group of layer-specific network protocols.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--OM8E6Cex--/c_limit,f_auto,fl_progressive,q_auto,w_880/https:/cdn-images-1.medium.com/max/2000/0*zwEyy3HRra-j5a4G.png)

For the purpose of this application, we need not concern ourselves with many of the lower-level protocols. But we do need to know that we are using something called the Transmission Control Protocol(TCP).

TCP and UDP are transport-layer protocols — they govern how data is sent from one point to another. We are building on top of TCP, which means we do not need to care about how data is sent, only what and where data is sent.

The main difference between TCP and UDP is that TCP guarantees reliable delivery, without any information lost, duplicated, or out-of-order. UDP does not guarantee the same and leaves it up to the application layer to handle dropped packets. It requires the server to acknowledge receipt of data.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--GIB1ZRi8--/c_limit,f_auto,fl_progressive,q_auto,w_880/https:/cdn-images-1.medium.com/max/2000/0*3UKurt9qle5Wg1AC.png)

UDP is normally used for time-sensitive transmissions where a dropped packet is preferred to waiting for lost packets to be re-transmitted. Imagine a real-time voice call, for example. If a packet has been lost, it won’t make sense to wait for that piece of audio to arrive because, by the time it does, it would already be too old. Furthermore, you would probably still be able to piece the conversation together without that packet of audio.

TCP, on the other hand, would have kept stubbornly resending that piece of lost audio even though it is already too old to be of any use. However, for most other applications, this aspect of TCP is extremely valuable. In our chat application, we don’t want to have to deal with lost packets because we always want to receive complete messages without any errors.

* Flask

# Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

# Requirements

# Flask==0.10.1

# 

# Flask-SocketIO

# 

# Jinja2==2.7.2

# 

# MarkupSafe==0.18

# 

# Werkzeug==0.9.4

# 

# gevent==1.0

# 

# gevent-socketio==0.3.6

# 

# gevent-websocket==0.9.2

# 

# greenlet==0.4.2

# 

# itsdangerous==0.23

# 

# ujson==1.33

# Working

* Socket Programming

Sockets can be thought of as endpoints in a communication channel that is bi-directional and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server-side. Any client that has a socket associated with the same port can communicate with the server socket.

* Multi-Threading

A thread is a sub-process that runs a set of commands individually of any other thread. So, every time a user connects to the server, a separate thread is created for that user, and communication from the server to the client takes place along individual threads based on socket objects created for the sake of the identity of each client.

We will require two scripts to establish this chat room. One to keep the serving running, and another that every client should run in order to connect to the server.

* Server Side Script

The server-side script will attempt to establish a socket and bind it to an IP address and port specified by the user (windows users might have to make an exception for the specified port number in their firewall settings, or can rather use a port that is already open). The script will then stay open and receive connection requests and will append respective socket objects to a list to keep track of active connections. Every time a user connects,

a separate thread will be created for that user. In each thread, the server awaits a message and sends that message to other users currently on the chat. If the server encounters an error while trying to receive a message from a particular thread, it will exit that thread.

* Usage

This server can be set up on a local area network by choosing any on the computer to be a server node, and using that computer’s private IP address as the server IP address.

For example, if a local area network has a set of private IP addresses assigned ranging from 192.168.1.2 to 192.168.1.100, then any computer from these 99 nodes can act as a server, and the remaining nodes may connect to the server node by using the server’s private IP address. Care must be taken to choose a port that is currently not in usage. For example, port 22 is the default for ssh, and port 80 is the default for HTTP protocols. So these two ports preferably, shouldn’t be used or reconfigured to make them free for usage.

However, if the server is meant to be accessible beyond a local network, the public IP address would be required for usage. This would require port forwarding in cases where a node from a local network (node that isn’t the router) wishes to host the server. In this case, we would require any requests that come to the public IP addresses to be re-routed towards our private IP address in our local network, and would hence require port forwarding.

* Client-Side Script

The client-side script will simply attempt to access the server socket created at the specified IP address and port. Once it connects, it will continuously check as to whether the input comes from the server or from the client, and accordingly redirects output. If the input is from the server, it displays the message on the terminal. If the input is from the user, it sends the message that the user enters to the server for it to be broadcasted to other users.

**Code**

****

****

****

****

**Output**

