REMOTE FILE PROVIDER USING TELEGRAM BOT

MAJOR PROJECT REPORT

Submitted in partial fulfilment

To complete 6th Semester

Of

BACHELOR OF COMPUTER APPLICATIONS

By

SHARMON CHAHAL

(Enrolment Number: 03621202018)

Department of Computer Science

Maharaja Surajmal Institute

C – 4, Janak-Puri, New Delhi - 110058

TO WHOMESOEVER IT MAY CONCERN

This is to certify that **SHARMON CHAHAL** has developed **"REMOTE FILE PROVIDER USING TELEGRAM BOT**" under my guidance at **MAHARAJA SURAJMAL INSTITUTE** from **JANUARY – July, 2021** for the partial fulfillment to complete 6th semester of **"Bachelor of Computer Applications"**. This is his original work.

Signature

Dr. MENAL DAHIYA

(Associate Professor)
Dept. Of Compute Science
Maharaja Surajmal Institute
C – 4, Janak-Puri, New Delhi

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

External examiner

Dr. MENAL DAHIYA
(Associate Professor)

Dept. Of Compute Science

Maharaja Surajmal Institute

C – 4, Janak-Puri, New Delhi

CANDIDATE'S DECLARATION

I **SHARMON CHAHAL** with enrollment number **03621202018**, hereby declare that the work which is being presented in the project report entitled "REMOTE FILE PROVIDER USING TELEGRAM BOT FOR MAJOR PROJECT REPORT" is in partial fulfillment of the requirement to complete the 6th semester of "Bachelor of Computer Applications" submitted in **Maharaja Surajmal Institute, C – 4, Janak-Puri, New Delhi – 58**, is an authentic record of my work carried out during the period from April-July under the guidance of **Dr. Menal Dahiya** Associate Professor, Department of Computer Science, Maharaja Surajmal Institute

The matter embodied in this report has not been submitted by me for the award of any other degree.

SHARMONCHAHAL
Enrollment Numbers- 03621202018
B.C.A. 6th SEM
Department of Computer Science
Maharaja Surajmal Institute
C – 4, Janak-Puri, New Delhi – 58

ACKNOWLEDGEMENT

I WOULD LIKE TO EXPRESS MY SPECIAL THANKS OF GRATITUDE TO MY TEACHER **DR. MENAL DAHIYA** AS WELL AS OUR DIRECTOR WHO GAVE ME THE GOLDEN OPPORTUNITY TO DO THIS WONDERFUL PROJECT ON THE TOPIC REMOTE FILE PROVIDER USING TELEGRAM BOT, 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 THEM.

SECONDLY, I WOULD ALSO LIKE TO THANK MY PARENTS AND FRIENDS WHO HELPED ME A LOT IN FINALIZING THIS PROJECT WITHIN THE LIMITED TIME FRAME.

SHARMON CHAHAL

ABSTRACT

As we all know Internet bots, also known as web robots, WWW robots or simply bots, are software applications that perform repetitive tasks automatically or on a schedule over the internet, tasks that would be too mundane or time-consuming for an actual person. Search engines use them to surf the web and methodically catalogue information from websites, trading sites make them look for the best bargains in seconds, and some websites and services employ them to deliver important information like weather conditions, news and sports, currency exchange rates.

In this project we use a combination of a local web server using raspberry pi containing the required files which can be sent through telegram and the file size limit is up to 2GB.

INTRODUCTION

In this work, we take on the role of making a basic REMOTE FILE PROVIDER USING TELEGRAM BOT to automate the basic process of accessing files without the hassle of accessing your personal computer again and again and without the worry of getting your external hard drive stolen with this you can simply give a pre-defined command to the bot and you will get your file delivered as Telegram is multi-platform so it will be easy to access your bot regardless of the device

So before the code begins a short overview of the bots is given
A bot is a software application that is programmed to do certain tasks. Bots are automated, which means they run according to their instructions without a human user needing to manually start them up every time. Bots often imitate or replace a human user's behaviour. Typically they do repetitive tasks, and they can do them much faster than human users could.
Bots usually operate over a network; more than half of Internet traffic is bots scanning content, interacting with webpages, chatting with users, or looking for attack targets. Some bots are useful, such as search engine bots that index content for search or customer service bots that help users. Other bots are "bad" and are programmed to break into user accounts, scan the web for contact information for sending spam, or perform other malicious activities. If it's connected to the Internet, a bot will have an associated IP address.

Bots can be:

Chat-bots: Bots that simulate human conversation by responding to certain phrases with programmed responses

Web crawlers (Google-bots): Bots that scan content on webpages all over the Internet

Social bots: Bots that operate on social media platforms

Malicious bots: Bots that scrape content, spread spam content, or carry out credential stuffing attacks

CODE OF "THE REMOTE FILE PROVIDER USING TELEGRAM BOT"

Step 1-

The first step is to **import all the libraries**, here we will obviously need the teleport library to use the Telegram bot. We also make use of the time, time date library to read the current time for Raspberry pi. Then we create an object now in which the value is stored.

main.py

```
import time, datetime
import telepot
from telepot.loop import MessageLoop

now = datetime.datetime.now()
```

Step 2-

The next step is to **create a function for taking actions based on incoming commands** from Telegram app on Mobile. Here the name of the function is action. It is inside this function where the bot comes to life. Our bot cannot initiate a conversation on its own, it can only reply if we ask something. So each time we ask something there will be chat id. This chat id is something similar to a address, only using this chat id a bot can reply back to us. So the first step is to read the chat id and the message it is trying to say to us. We also print the received message for debugging purpose.

```
def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print('Received: %s' % command)
```

Step 3-

Further down inside the function we compare this command with a predefined text and perform particular tasks. This first command will be ***/hi*** to which we reply

```
if command == '/hi':
    telegram_bot.sendMessage(chat_id, str('Hi! I AM MPR PROJECT'))
    telegram_bot.sendMessage(chat_id, str('Use /time to aske me the current time '))
    telegram_bot.sendMessage(chat_id, str('Use /logo for the college logo to be sent '))
    telegram_bot.sendMessage(chat_id, str('LIST OF NOTES AND BOOKS YOU CAN ASK FROME ME'))
    telegram_bot.sendMessage(chat_id, str('1 Python101 book --(/py101)'))  # (/py101 command to acess the file)
    telegram_bot.sendMessage(chat_id, str('2 Python202 book --(/py202)'))  # (/py202 command to acess the file)
    telegram_bot.sendMessage(chat_id, str('3 Basic Networking notes --(/netw)'))
    telegram_bot.sendMessage(chat_id, str('4 C++ Notes --(/c1)'))
```

Step 4-

The next command will be ***/time***, to which we reply the current time. We already have the time and date in now, here simply split it based on hour and minute and add it as using the *str* function.

```
elif command == '/time':
    telegram_bot.sendMessage(chat_id, str(now.hour) + str(":") + str(now.minute))
```

The next command will be ***/logo***, to which the bot will fetch an image from a URL and send it to us. An image can be sent either from a URL or from the hard disk. Here I have just used the URL which displays the logo of our **COLLEGE**

elif command == '/logo':telegram_bot.sendPhoto (chat_id, photo="https://i.pinimg.com/avatars/circuitdigest_1464122100_280.jpg")

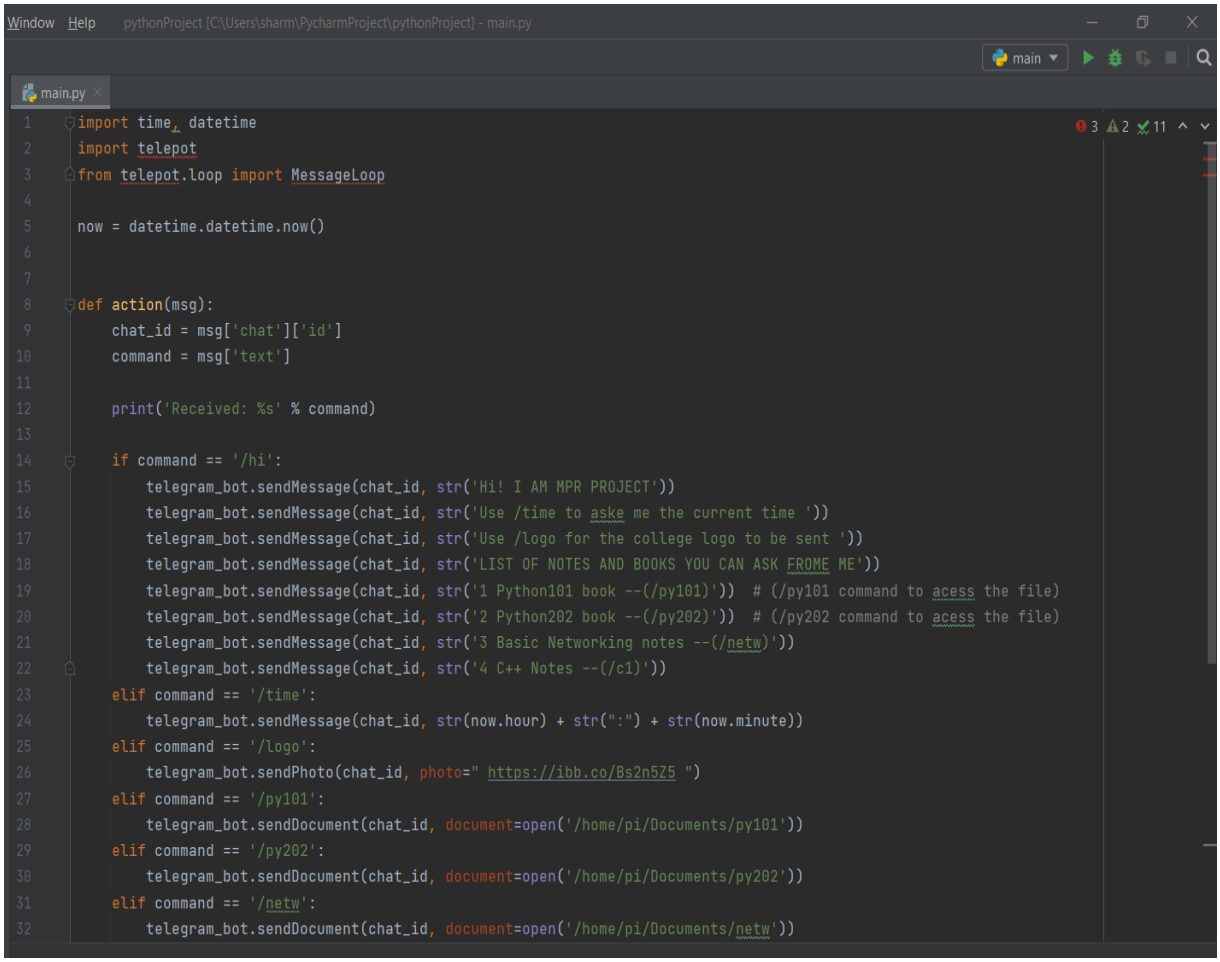
The next command will be ***/py101***, which will send the file named py101 from the hard disk. You can send any file that you wish to by changing the address of the directory and specifying the file

```
elif command == '/py101':
    telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/py101'))
elif command == '/py202':
    telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/py202'))
elif command == '/netw':
    telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/netw'))
elif command == '/c1':
    telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/c1'))
```

Okay now comes the **most important step,** this is where we give access of our Telegram bot to the Python script. Here we name out bot as *telegram_bot* and assign it the token address that was given by our bot-father in step 3. . We also use the print get me to display the details of the Bot on the shell screen, this will help us notice things working.

```
telegram_bot = telepot.Bot('1817933623:AAHA2ZXYkdotuxPtHLBYhCfepxKELRPOhzw')
print(telegram_bot.getMe())
```

FINAL CODE

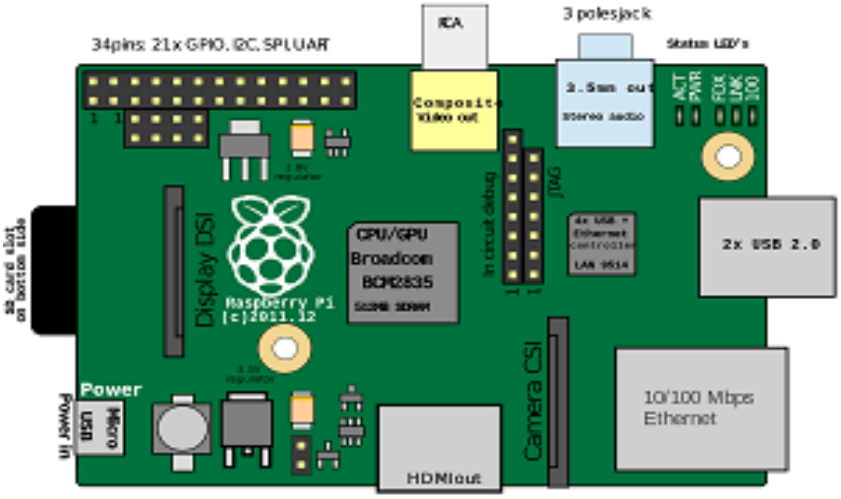


```
import time, datetime
import telepot
from telepot.loop import MessageLoop

now = datetime.datetime.now()


def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print('Received: %s' % command)

    if command == '/hi':
        telegram_bot.sendMessage(chat_id, str('Hi! I AM MPR PROJECT'))
        telegram_bot.sendMessage(chat_id, str('Use /time to aske me the current time '))
        telegram_bot.sendMessage(chat_id, str('Use /logo for the college logo to be sent '))
        telegram_bot.sendMessage(chat_id, str('LIST OF NOTES AND BOOKS YOU CAN ASK FROME ME'))
        telegram_bot.sendMessage(chat_id, str('1 Python101 book --(/py101)'))  # (/py101 command to acess the file)
        telegram_bot.sendMessage(chat_id, str('2 Python202 book --(/py202)'))  # (/py202 command to acess the file)
        telegram_bot.sendMessage(chat_id, str('3 Basic Networking notes --(/netw)'))
        telegram_bot.sendMessage(chat_id, str('4 C++ Notes --(/c1)'))
    elif command == '/time':
        telegram_bot.sendMessage(chat_id, str(now.hour) + str(":") + str(now.minute))
    elif command == '/logo':
        telegram_bot.sendPhoto(chat_id, photo=" https://ibb.co/Bs2n5Z5 ")
    elif command == '/py101':
        telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/py101'))
    elif command == '/py202':
        telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/py202'))
    elif command == '/netw':
        telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/netw'))
```

```
    elif command == '/c1':
        telegram_bot.sendDocument(chat_id, document=open('/home/pi/Documents/c1'))


telegram_bot = telepot.Bot('1817933623:AAHA2ZXYkdotuxPtHLBYhCfepxKELRPOhzw')
print(telegram_bot.getMe())

MessageLoop(telegram_bot, action).run_as_thread()
print('Up and Running....')

while 1:
    time.sleep(10)
```

TECHNOLOGIES USED

1- RASPBERRY PI (IOT)



Raspberry Pi is a low-cost, basic computer that was originally intended to help spur interest in computing among school-aged children. The Raspberry Pi is contained on a single circuit board and features ports for:

- HDMI
- USB 2.0
- Composite video
- Analogue audio
- Power
- Internet
- SD Card

The computer runs entirely on open-source software and gives students the ability to mix and match software according to the work they wish to do.

The Raspberry Pi debuted in February 2012. The group behind the computer's development - the Raspberry Pi Foundation - started the project to make computing fun for students, while also creating interest in how computers work at a basic level. Unlike using an encased computer from a manufacturer, the Raspberry Pi shows the essential guts behind the plastic. Even the software, by virtue of being open-source, offers an opportunity for students to explore the underlying code - if they wish.

The Raspberry Pi is believed to be an ideal learning tool, in that it is cheap to make, easy to replace and needs only a keyboard and a TV to run. These same strengths also make it an ideal product to jumpstart computing in the developing world.

But raspberry pi alone is not enough it’s just a board it requires a SD card in which OS is installed, a power source, a case to protect the pi as a single static shock from our hand can ruin the board, and a micro HDMI to HDMI cable to connect to the display/TV , a SD card reader, heat sinks and fans to prevent overheating of the processors.

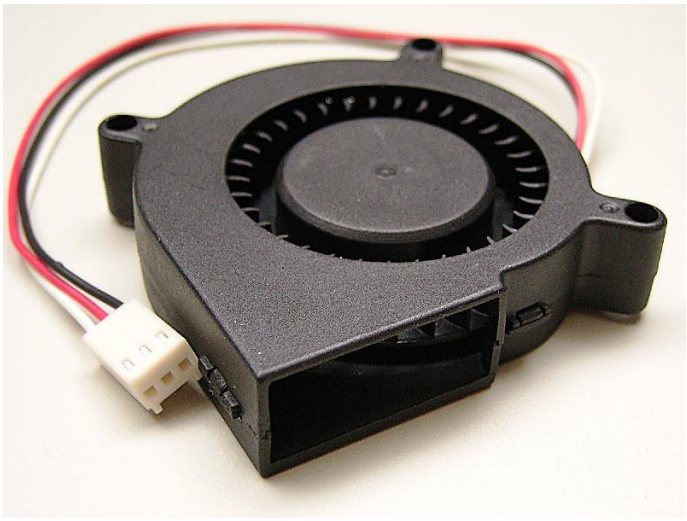
2. HEAT SINKS

A heat sink is a component that increases the heat flow away from a hot device. It accomplishes this task by increasing the device's working surface area and the amount of low-temperature fluid that moves across its enlarged surface area.
Heat sinks are most commonly utilized in active, passive or hybrid configurations.

Passive heat sinks rely on natural convection, meaning the buoyancy of hot air alone causes the airflow generated across the heat sink system. These systems are advantageous as they do not require secondary power or control systems to remove heat from the system. However, passive heat sinks are less effective at transferring heat from a system than active heat sinks.

Active heat sinks utilize forced air to increase fluid flow across the hot area. Forced air is most commonly generated by a fan, blower, or even movement of the entire object—such as a motorcycle's engine being cooled by the air passing along the heat sink fins designed into the engine. One example of a fan producing forced air across a heat sink is the fan in your personal computer turning on after your computer gets warm. The fan forces air across the heat sink, which allows more unheated air to move across the heat sink surface, thus increasing the total thermal gradient across the heat sink system and allowing more heat to exit the overall.

3. COOLING FAN

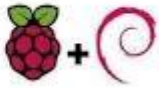


To cool the components of raspberry pi, fans are used to move heated air away from the components and draw cooler air over them. Fans attached to components are usually used in combination with a heat sink to increase the area of heated surface in contact with the air, thereby improving the efficiency of cooling. Fan control is not always an automatic process. A computer's BIOS (basic input/output system) can control the speed of the built-in fan system for the computer.

They are very important for the functioning of our IOT as if the board overheats there is a chance of the board getting fried and we might lose our IOT.

4. MEMORY CARD

The Raspberry Pi should work with any compatible SD card, although a calss 10 card is a must.



Raspberry pi requires a class 10 memory card as the OS is installed in it.

The card class determines the sustained write speed for the card; a class 4 card will be able to write at 4MB/s, whereas a class 10 should be able to attain 10 MB/s. However, it should be noted that this does not mean a class 10 card will outperform a class 4 card for general usage, because often this write speed is achieved at the cost of read speed and increased seek times.

5. SD CARD READER

SD CARD READER is used to inset the memory card with the operating system installed into the raspberry pi.

6. 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

Python Syntax compared to other programming languages:

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

7. SSH

SSH or Secure Shell is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

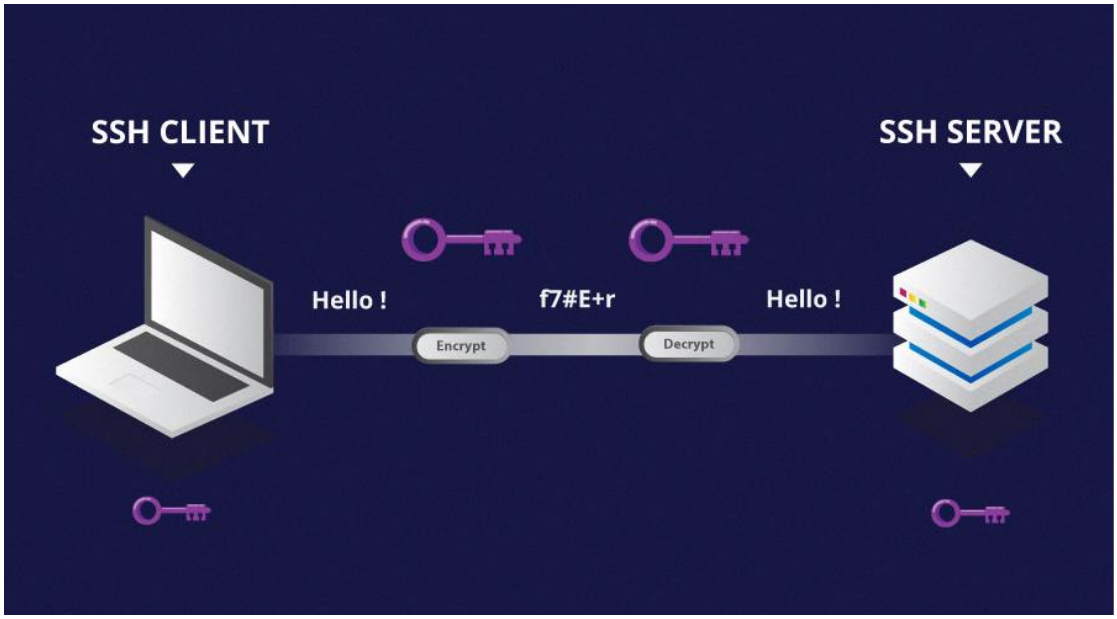
SSH provides a secure channel over an unsecured network by using a client–server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access UNIX-like operating systems, but it can also be used on Microsoft Windows. Windows 10 uses OpenSSH as its default SSH client and SSH server.

Despite popular misconception, SSH is not an implementation of Telnet with cryptography provided by the Secure Sockets Layer (SSL).

SSH was designed as a replacement for Telnet and for unsecured remote shell protocols such as the Berkeley rsh and the related rlogin and rexec protocols. Those protocols send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet.

SSH uses public-key cryptography to authenticate the remote computer and allow it to authenticate the user, if necessary. There are several ways to use SSH, one is to use automatically generated public-private key pairs to simply encrypt a network connection, and then use password authentication to log on.

Another is to use a manually generated public-private key pair to perform the authentication, allowing users or programs to log in without having to specify a password. In this scenario, anyone can produce a matching pair of different keys (public and private). The public key is placed on all computers that must allow access to the owner of the matching private key (the owner keeps the private key secret). While authentication is based on the private key, the key itself is never transferred through the network during authentication. SSH only verifies whether the same person offering the public key also owns the matching private key. In all versions of SSH it is important to verify unknown public keys, i.e. associate the public keys with identities, before accepting them as valid. Accepting an attacker's public key without validation will authorize an unauthorized attacker as a valid user.



CONCLUSION

In this project we have made a feasible solution to the problem of file sharing and privacy .Using this we don't have to worry about the files getting stolen nor do we have to invest a large amount for security of files we rely on security of telegram rather than making it ourselves we just have to make a file directory and assign the commands to the bot

BIBLIOGRAPHY

1. https://en.wikipedia.org/wiki/Python_(programming_language)#:~:text=Python%20was%20conceived%20in%20the,implementation%20began%20in%20December%201989.
2. https://en.wikipedia.org/wiki/Twitter
3. https://www.pythonanywhere.com/user/consoles/20652499/
4. https://core.telegram.org/bots/