DatumNode

A Secure Data Repository

By

NC Ahmed Rauf

NC Ayesha Gillani

NC Shaikh Mohammad Zain

NC Muhammad Usman

Project Supervisor

AP Waleed Bin Shahid

Submitted to the Faculty of Department of Electrical Engineering, Military College of Signals, National University of Sciences and Technology, Islamabad in partial fulfilment for the requirements of a B.E. Degree in Electrical (Telecom) Engineering

July 2020

CERTIFICATE OF CORRECTIONS & APPROVAL

Certified that work contained in this thesis titled “DatumNode (A secure data repository), carried out by NC Ahmed Rauf, NC Ayesha Gillani, NC Shaikh Mohammad Zain, and NC Muhammad Usman under the supervision of AP Waleed Bin Shahid for partial fulfillment of Degree of Bachelors of Telecommunication Engineering, in Military College of Signals, National University of Sciences and Technology during the academic year 2019-2020 is correct and approved. The material that has been used from other sources it has been properly acknowledged / referred.

Approved By

Supervisor

AP Waleed Bin Shahid

IS Dept, MCS

Dated:

DECLARATION

No portion of work presented in this thesis has been submitted in support of another award or qualification in either this institution or anywhere else.

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student

Registration Number

Signature of Supervisor

ACKNOWLEDGEMENTS

We are thankful to our Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in our mind to improve it. Indeed, We could have done nothing without Your priceless help and guidance. Whosoever helped us throughout the course of our thesis, whether our parents or any other individual was Your will, so indeed none be worthy of praise but You. We are profusely thankful to our beloved parents who raised us when we were not capable of walking and continued to support us throughout in every department of our life. We would also like to express special thanks to our supervisor AP. Waleed Bin Shahid for his help throughout our thesis and also for the Network Security courses he has taught us. We can safely say that we haven't learned any other technical course in such depth than the one which he has taught. Finally, we would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

DEDICATION

This research is lovingly dedicated to our parents, well-wishers and most of all our supervisor AP Waleed Bin Shahid, without whose unstinting cooperation and unflinching support, a work of this magnitude would not have been possible.

ABSTRACT

Nowadays, data breaches that affect hundreds of millions of people have become far too common. Most of us are under data surveillance in some form. Although there are some regulatory safeguards in place to protect users' privacy, it's hard to say whether these are enough. Businesses spend millions on storage providers in order to securely and privately store their data. ‘DatumNode’ provides a solution to this critical problem for all privacy-conscious people and smaller organizations. Whether you are an android user or a pc user, DatumNode helps you protect files, photos, media, contacts, and other important data by storing it on a device of your own. It would also offer a personalized email domain, a secure chat between consumers, and can also act as a VPN to further protect users’ online activity. Furthermore, all data on the device is encrypted and immutable unless the user desires otherwise. 'DatumNode’ would basically appear as a one-time reasonable investment and secure substitute for these free storage services so that only the user controls his/her data. ’DatumNode’ can also support a limited number of multiple users in order to make it useful for smaller organizations. It would be a hardware device with large storage and processing capacity along with a network interface that can be accessed locally or publicly.

TABLE OF CONTENTS

ABSTRACT **7**

1: Introduction **12**

1.1 Problem Statement 12

1.2 Data loss and its Financial Impact 12

1.3 Project Overview 13

1.4 Scope of Project 14

1.5 Organization of Document 14

2: Literature Review **16**

2.1 Literature review 16

2.2 Existing Solutions and their flaws 16

2.3 Novelty of DatumNode: 17

3: Technological Requirements **20**

3.1 Hardware 20

3.1.1 Raspberry Pi 3 20

3.1.2 LCD screen 21

3.1.3 Power bank 21

3.2 Software Requirements 22

3.2.1 Python 22

3.2.2 Android Studio 23

3.3 Operating System Requirements 23

3.4 Setting up Working Environment 24

3.4.1 Installing OS 24

3.4.2 Installing software for creating application 24

3.4.3 Installing NGROK 24

3.4.4 Setting up NGROK 26

3.4.5 Configuring VPN 27

3.4.6 Setting up Email Server ... 28

3.4.8 Setting up Flask API ... 29

4: Methodology ... 32

4.1 Architecture of DatumNode ... 32

4.2 Application overview ... 33

4.2.1 Desktop Application overview ... 33

4.2.2 Android Application overview ... 34

4.3 Registration on Desktop ... 34

4.3.1 Registration on Android ... 35

4.4 Encryption ... 36

4.5 Hashing ... 38

4.6 Desktop App Walkthrough ... 39

4.7 Android App Walkthrough ... 43

4.8 Using OVPN on the client side ... 46

4.9 Sending & Receiving Emails ... 48

5: Results & Analysis ... 51

5.1 Project Block Diagram ... 51

5.2 Initiation ... 51

5.3 Objectives ... 52

6: Conclusion & Future work ... 54

6.1 Conclusion ... 54

6.2 Future Work ... 55

6.2.1 Software Enhancements ... 55

6.2.2 Hardware Enhancements ... 55

References ... 56

Appendix A: Synopsis Form – DatumNode ... 58

LIST OF FIGURES

Figure 4. 1 DatumNode hardware design ____ 20
Figure 4. 2 NGROK [8] ____ 25
Figure 4. 3 NGROK help terminal ____ 26
Figure 4. 4 Setting up NGROK ____ 26
Figure 4. 5 Downloading OpenVPN ____ 27
Figure 4. 6 Configuring OpenVPN ____ 27
Figure 4. 7 Architecture of DatumNode ____ 32
Figure 4. 8 Desktop App overview ____ 33
Figure 4. 9 Android App overview ____ 34
Figure 4. 10 Dialog box ____ 35
Figure 4. 11 Android Registration Window ____ 35
Figure 4. 12 Encryption Pseudo code ____ 37
Figure 4. 13 Encryption in Flask API ____ 37
Figure 4. 14 Hashing Pseudo code ____ 38
Figure 4. 15 Creating secure TCP connection ____ 39
Figure 4. 16 DatumNode login page ____ 39
Figure 4. 17 DatumNode UI ____ 40
Figure 4. 18 DatumNode sync window ____ 40
Figure 4. 19 DatumNode retrieval window ____ 41
Figure 4. 20 Client’s chat application window ____ 41
Figure 4. 21 Android Login Window ____ 43
Figure 4. 22 Android Uploaded images Window ____ 44
Figure 4. 23 Android Gallery to choose files [15] ____ 44
Figure 4. 24 Android Uploaded images window with log out button ____ 45
Figure 4. 25 Making client.ovpn available to the client ____ 46
Figure 4. 26 Downloading client.ovpn ____ 46
Figure 4. 27 Downloading OpenVPN ____ 47
Figure 4. 28 Importing client.ovpn on OpenVPN Connect ____ 47
Figure 4. 29 Connection established ____ 48
Figure 4. 30 DatumNode mail login screen ____ 48
Figure 4. 31 DatumNode mail inbox ____ 49
Figure 4. 32 Block diagram ____ 51

CHAPTER 1: INTRODUCTION

1: Introduction

This chapter provides comprehensive introduction of the project titled “DatumNode”

1.1 Problem Statement

The ubiquity of internet and mobile services has produced enormous amount of data, not just for corporates but for individuals as well. This mammoth use of internet is basically triggered by the use of smartphones [1]. Nearly 3 Billion people use smartphones across the globe while 21 percent of Pakistan’s nearly 220 million population rely heavily on mobile internet services [2]. This phenomenon gave rise to a lot of storage issues at the user end thereby giving rise to the cloud technology whereby data and services are hosted by the third party. A lot of service providers, especially the big technology giants offer storage and processing services and now our entire digital life (photos, emails, calendar, files, videos etc.) resides with them. In this case, user is not the only one having control over his data and resources, thereby breeding a bunch of trust issues for everyone who is privacy conscious and does not want these big companies to spy on them.

1.2 Data loss and its Financial Impact

Cyber security research and reports from the past several years indicate that cyber-attacks are rapidly increasing, and the number of new attacks will surpass those of previous years. The average cost of cybercrime for an organization is estimated to be **$13 million per year,** according to Accenture's global study.

Here are some cyber security attacks that were reported in 2018 and 2019[1]

- ***1.16 billion email addresses and passwords exposed***

  The number of “unique combinations of email addresses and passwords” that was discovered in 2019 in a massive breach called “Collection 1.” This load of information was discovered by an IT security researcher and is thought to be the largest breach in history to date, according to an article by Fortune.

- ***Ransomware damage estimated to reach $20 billion globally***

The Cybersecurity Ventures annual crime report indicates that the same costs will reach $11.5 billion annually this year and $20 billion per year by 2021. Unsurprisingly, this type of year-over-year increase in anticipated damages makes ransomware the fastest-growing type of cybercrime in the past year.

- ***38.4% of mining industry users receive malicious emails***

According to Symantec's Internet Security Threat Report (ISTR) 2019 report, 38.4% of users in the mining industry were targeted with malicious emails.

- ***1 in 302 emails targeting public administration users are malicious***

Email users working in the public administration sector receive one malicious email for every 302 emails they receive, according to Symantec's ISTR 2019 report.

- ***Ransomware attacks to increase 5X by 2021***

The Cybersecurity Almanac 2019 from Cybersecurity Ventures estimates that ransomware attacks against healthcare organizations will increase by this amount between 2017 and 2021.

1.3 Project Overview

'DatumNode' is a hardware solution large storage and processing capacity along with a network interface that can be accessed over the internet or via data cable (in case of device proximity). It protects files, photos, media, contacts and other important data by storing it on a device of your own. It would also offer a personalized email domain and can act as a VPN to further protect users' online activity. DatumNode would basically replace these free storage services so that only the user will have access and complete control over his data. DatumNode can also support limited number of multiple users in order to make it useful for smaller organization.

1.4 Scope of Project

To develop a secure personal server to securely store and protect your personal information so that only the user has access to it. Encrypting all data stored on the device and messages while communicating using Chat app. Lastly, to create a Personalized VPN and hosting an email server. The overall scope of this project includes, but is not limited to:

- Computer Networks (TCP/IP Architecture)
- Network Security (Encryption, VPNs)
- Programming (Sockets, Web, Android, File System)
- Hardware (Interfacing, Cable and Wi-Fi connectivity, SSD)
- Operating System (Windows, Android, Ubuntu)

1.5 Organization of Document

Chapter 1	Provides overview and problem definition of DatumNode
Chapter 2	Deals with the literature review carried out for DatumNode and discusses its core features.
Chapter 3	Discusses the hardware and software requirements of DatumNode, including technical specifications and setup for working environment
Chapter 4	Discusses the methodology followed by DatumNode in detail.
Chapter 5	Provides all results, testing and analysis regarding the main working of DatumNode
Chapter 6	Presents overview of enhancements and future work

CHAPTER 2: LITERATURE REVIEW

2: Literature Review

This chapter deals with the literature review and drawbacks of using different cloud services offered worldwide, the existing solutions and their flaws, and the novelty of our solution.

2.1 Literature review

Relying on public cloud services like Google, Microsoft etc. comes with a price as our entire digital life resides there. With the growing number of breaches and privacy exploits associated with the continual evolution of IT infrastructure, organizations now require more privacy and control over their digital lives. DatumNode provides you the services to securely store and manage useful and critical data on a device you own. It comes with a secure chat, personalized email server and a VPN application as well. This hardware device can be placed anywhere the user wants let it be his home or workplace. It is an all-in-one product that is ideal for all those privacy conscious users. The product can be customized and comes with a dedicated hardware with good storage and processing.

2.2 Existing Solutions and their flaws

TOOL	DESCRIPTION	OS Supported	Cons
Helm	Helm is a personal server that stores your files and photos in it. Sends and receives emails from your custom domain. However, it only has 120 GB of storage space and your data backup is stored in their off-site personal server.	Android	• Expensive • Limited Storage • Restricted to Android App

pCloud	pCloudprovides 2TB storage.Ithas unlimited remote upload traffic. Its desktop application is slow.	Windows	• Slow data rate retrieval • Not personalized • No mobile support.
Google Drive	Most used cloud storage service that is quite user friendly and easy to use.	Windows/Linux/ Android	• Gets expensive when more space. • Data surveillances • Incomplete custody of data
Tonido	Tonidoserver allows you to access all your files from computer and smartphone. However, it is mainly for people that invest their data on a single computer rather than other devices.	Windows/Android	• Monthly payments • Data breach • Limited to single user

Table 2. 2 Existing Solutions and their flaws[2]

2.3 Novelty of DatumNode:

DatumNode is the Pakistan's first indigenous solution offering the following features:

- Unified communication
- Complete ownership
- Pocketable
- Automated and user- intuitive GUI

- For both Android/Windows users
- Extendable storage
- Cost Effective and Indigenous Scalable Solution

CHAPTER 3: TECHNOLOGICAL REQUIREMENTS

3: Technological Requirements

This chapter provides comprehensive details about technical requirements of DatumNode i.e. software, hardware and OS requirements.

3.1 Hardware

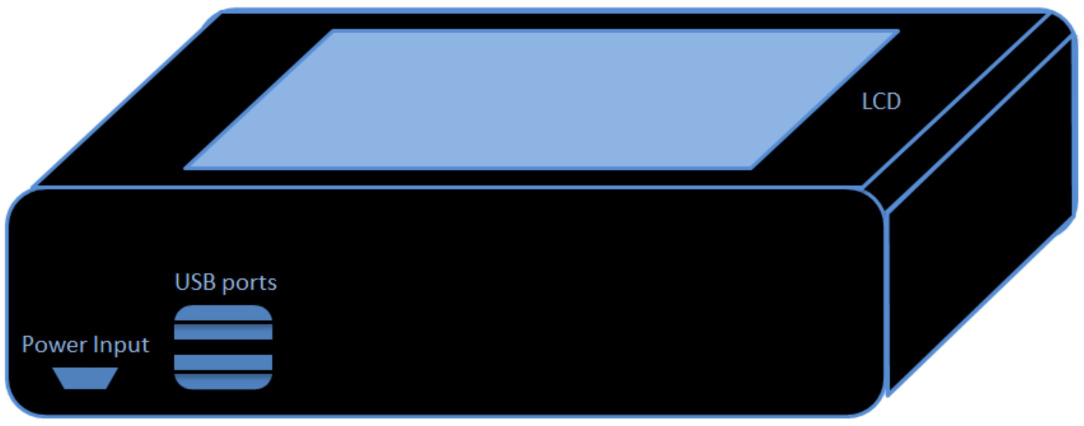
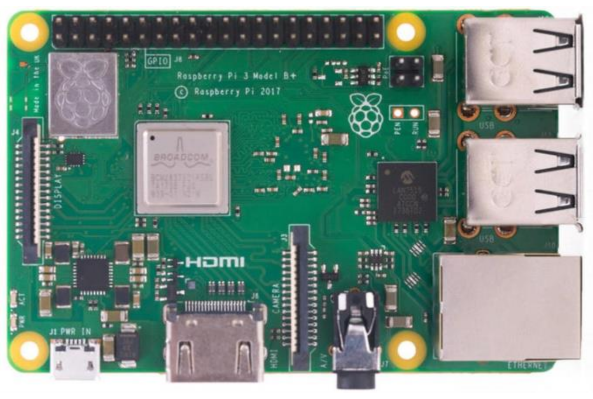


Figure 4. 1 DatumNode hardware design

The Hardware components required for the implementation of our Datumnode server include:

3.1.1 Raspberry Pi 3

A small single board PC that is convenient enough to carry around but at the same time powerful enough to carry out multiple tasks.



3.1.2 LCD screen

Interactive screen for users to visualize and interact with if needed. The screen is 3.5 inches diagonally (320 * 480 pixels)

3.1.3 Power bank

A 20000 mAH Qualcomm Quick Charge Anker power bank to keep the pi running when users are moving around.

3.2 Software Requirements

Datum Node's software requirements that have to be installed on the Pi are mentioned below

3.2.1 Python

Python v3.8 is utilized throughout this project. Since utilizing python for shell commands, SFTP and WEB API's are so simple, it would be most efficient for our project.

<u>The libraries used on the client side are mentioned below:</u>

- **Paramiko** – Paramiko is a Python (2.7, 3.4+) implementation of the SSHv2 protocol, providing both client and server functionality. It works by creating SSH tunnel to our DatumNode server
- **SQLite3** – SQLite3 is a very easy to use database engine. It is self-contained, serverless, zero-configuration and transactional. The Python Standard Library includes a module called "sqlite3" intended for working with this database. This module is a SQL interface compliant with the DB-API 2.0 specification. It creates databases to record user account details including hashes of each and every file synced with our server.
- **OS** – It commands to walk through our system locating files and returning their paths.
- **Tkinter** – It is a Python binding to the Tk GUI toolkit for interactive GUI
- **PyQt5** – PyQt5 is a comprehensive set of Python bindings for Qt v5 which is also used for interactive GUI
- **Hashib** – It is used to hashes and message digests by returning MD5 hash of chosen files.
- **PyAesCrypt** – It is a Python 3 file- encryption module and script that uses AES256-CBC for encrypting, decrypting before syncing and after retrieving each file.

- **Threading** – For carrying out background processes without interrupting the current ongoing processes.
- **Socket** – Initializing sockets to communicate with each other when binded together with an IP & port.

<u>Library used on the Server side mentioned below: (for mobile)</u>

- **Flask** – Handling post and get API requests from our android app users and giving Json responses for success or failure.
- **FlaskSQLalchemy** – Creating database objects for different individual entries in our android app.

3.2.2 Android Studio

- Flutter/DART – An open source UI development kit that we use to design our android app. The app is programmed to send out http requests to the Flask API on our server and await its responses.

3.3 Operating System Requirements

- 1.2 GHz clock frequency
- Up to 256 GB external SSD
- Up to 2 GB DDR2 RAM
- 4″ x 2″ designed box to host the LCD and Pi
- Wi-Fi
- 2 USB 3.0 ports

3.4 Setting up Working Environment

3.4.1 Installing OS

Raspbian Linux is installed into our Raspberry Pi through following steps [3]:

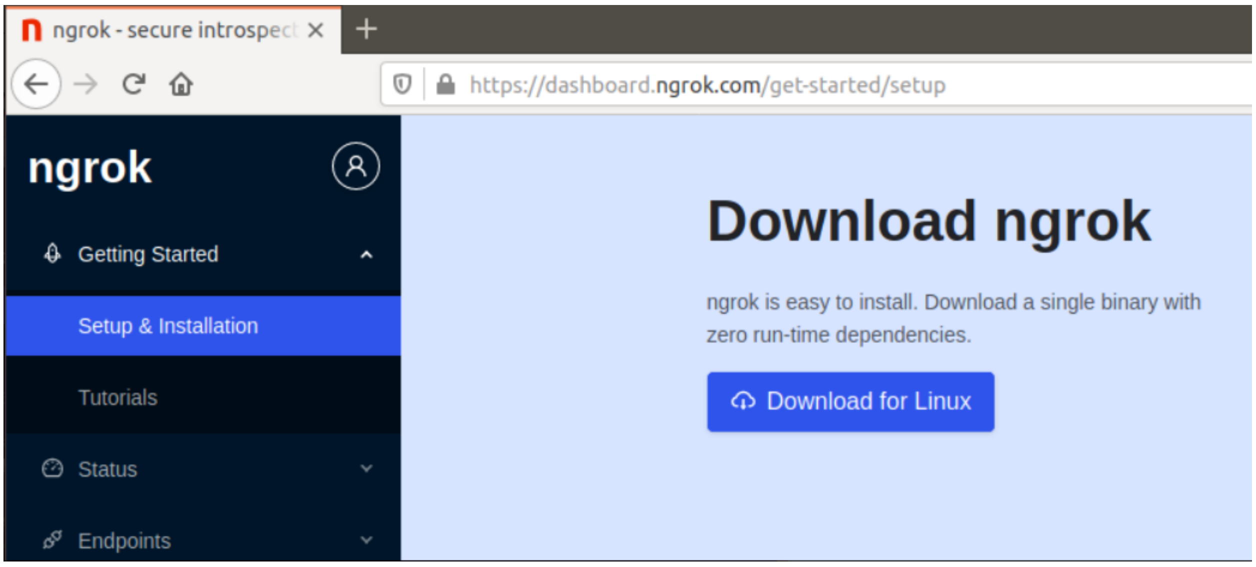
- Download Rasbian Image
- Format a USB and download Etcher
- Flash the USB with Rasbian Image using Etcher
- Enter USB in Raspberry Pi
- Once Raspberry Pi is turned on, sign in with the default username and password
- Create a new user
- Since Linux can run .py scripts by default, we install ‘pip’ to get libraries
- Install SSH (Secure Shell) server on the OS
- Enable SSH (Secure Shell)

3.4.2 Installing software for creating application

- Install PyQt5 using PIP command [4]
- Install Android studio [6]
- Install Flutter [7]

3.4.3 Installing NGROK

Step – 1 Search for NGROK on the Internet and then install it. Our server is Linux based so our NGROK file should be a Linux version.



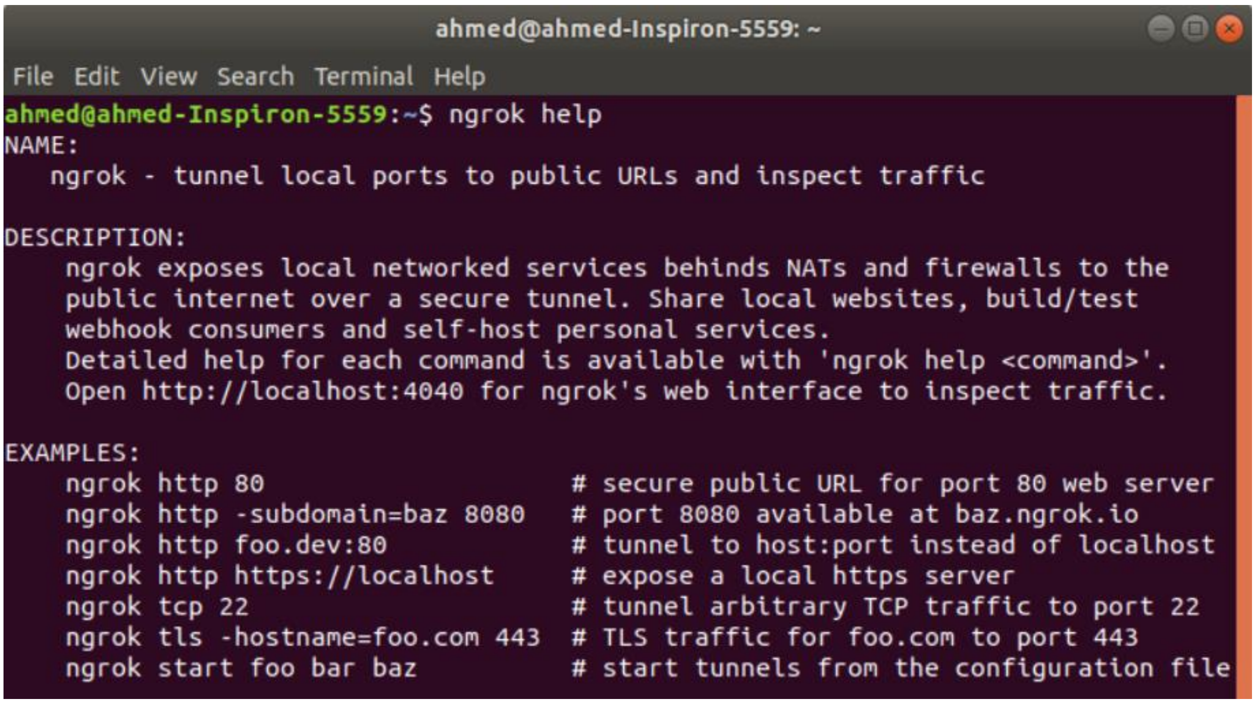
ngrok is easy to install. Download a single binary with zero run-time dependencies.

Figure 4. 2 NGROK [8]

Step – 2 Go to the download folder and then extract the NGROK zip file using command line or manually.

Step – 3 In the new version of NGROK we were told to add an authentication token for more features and longer session times. We used the command./ngrokauthtoken ****

Step – 4 Now after successfully installing NGROK, we can type in ngrok help to see all the possible port forwarding examples. We are mainly using tcp 22 for file sharing and https for email server.

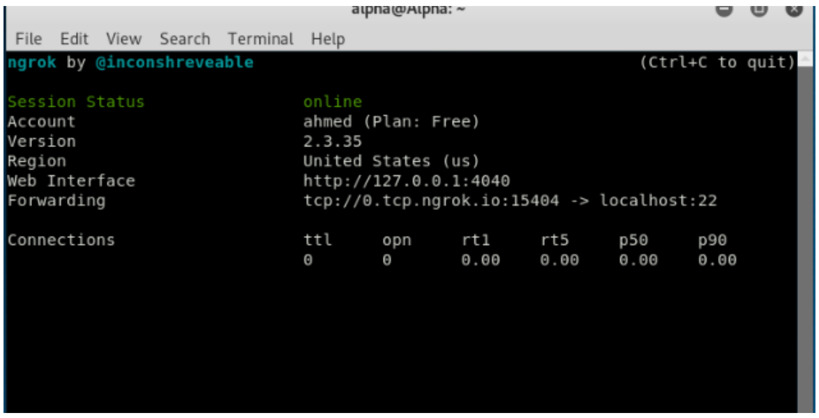

ahmed@ahmed-Inspiron-5559: ~
File Edit View Search Terminal Help

```
ahmed@ahmed-Inspiron-5559:~$ ngrok help
NAME:
   ngrok - tunnel local ports to public URLs and inspect traffic

DESCRIPTION:
    ngrok exposes local networked services behinds NATs and firewalls to the
    public internet over a secure tunnel. Share local websites, build/test
    webhook consumers and self-host personal services.
    Detailed help for each command is available with 'ngrok help <command>'.
    Open http://localhost:4040 for ngrok's web interface to inspect traffic.

EXAMPLES:
    ngrok http 80                    # secure public URL for port 80 web server
    ngrok http -subdomain=baz 8080   # port 8080 available at baz.ngrok.io
    ngrok http foo.dev:80            # tunnel to host:port instead of localhost
    ngrok http https://localhost     # expose a local https server
    ngrok tcp 22                     # tunnel arbitrary TCP traffic to port 22
    ngrok tls -hostname=foo.com 443  # TLS traffic for foo.com to port 443
    ngrok start foo bar baz          # start tunnels from the configuration file
```

Figure 4. 3 NGROK help terminal

3.4.4 Setting up NGROK

Step – 1 We run NGROK on port 22 of our DatumNode server



```
ngrok by @inconshreveable                                        (Ctrl+C to quit)

Session Status                online
Account                       ahmed (Plan: Free)
Version                       2.3.35
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    tcp://0.tcp.ngrok.io:15404 -> localhost:22

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

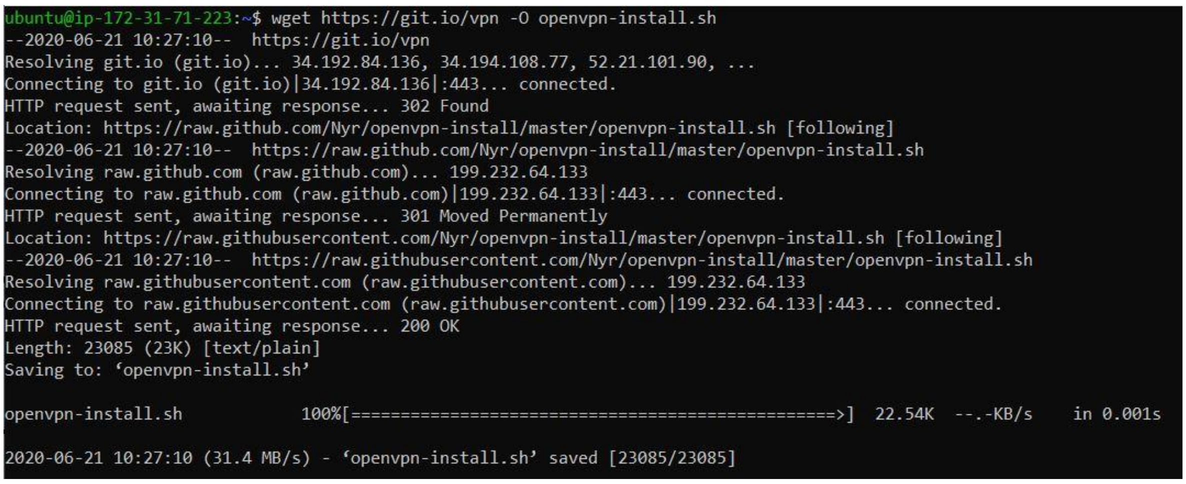
Figure 4. 4 Setting up NGROK

Our server is now forwarding port 15404 to our local host port 22. This will grant clients a public URL to connect to DatumNode. We can also use this locally by disabling NGROK and running it normally on local host.

3.4.5 Configuring VPN

The steps below are on the Server-side [10]

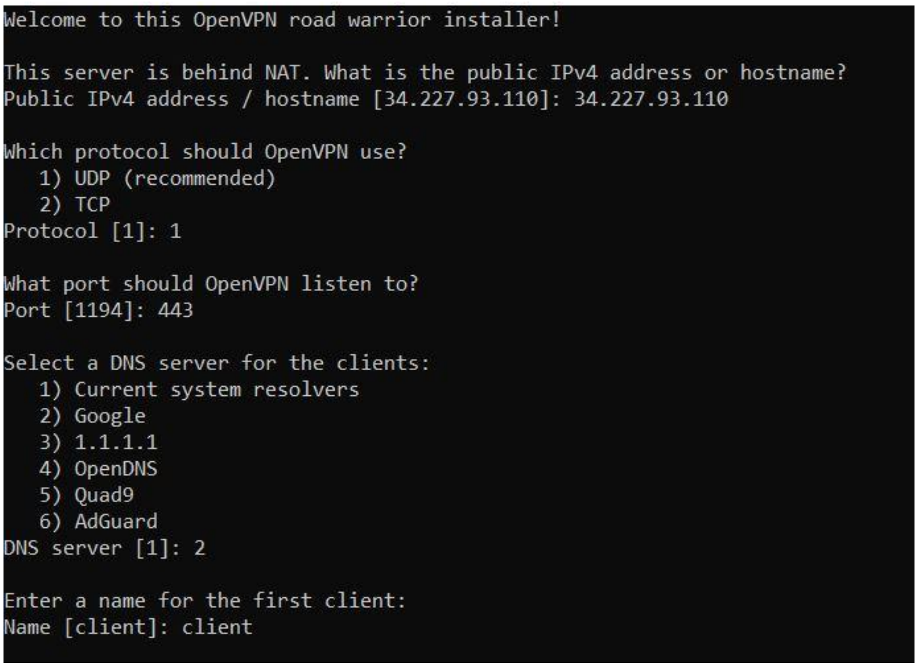
Step – 1 Downloading OpenVPN on the server side. We download the file from https://git.io/vpn. This would give us a file called 'open-vpn.sh'



```
ubuntu@ip-172-31-71-223:~$ wget https://git.io/vpn -O openvpn-install.sh
--2020-06-21 10:27:10--  https://git.io/vpn
Resolving git.io (git.io)... 34.192.84.136, 34.194.108.77, 52.21.101.90, ...
Connecting to git.io (git.io)|34.192.84.136|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.github.com/Nyr/openvpn-install/master/openvpn-install.sh [following]
--2020-06-21 10:27:10--  https://raw.github.com/Nyr/openvpn-install/master/openvpn-install.sh
Resolving raw.github.com (raw.github.com)... 199.232.64.133
Connecting to raw.github.com (raw.github.com)|199.232.64.133|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://raw.githubusercontent.com/Nyr/openvpn-install/master/openvpn-install.sh [following]
--2020-06-21 10:27:10--  https://raw.githubusercontent.com/Nyr/openvpn-install/master/openvpn-install.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.64.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.64.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23085 (23K) [text/plain]
Saving to: 'openvpn-install.sh'

openvpn-install.sh           100%[================================================>]  22.54K  --.-KB/s    in 0.001s

2020-06-21 10:27:10 (31.4 MB/s) - 'openvpn-install.sh' saved [23085/23085]
```

Figure 4. 5 Downloading OpenVPN

Step – 2 While configuring OpenVPN, we choose UDP protocol as it's recommended, and we forward port 1194 to 443.



```
Welcome to this OpenVPN road warrior installer!

This server is behind NAT. What is the public IPv4 address or hostname?
Public IPv4 address / hostname [34.227.93.110]: 34.227.93.110

Which protocol should OpenVPN use?
   1) UDP (recommended)
   2) TCP
Protocol [1]: 1

What port should OpenVPN listen to?
Port [1194]: 443

Select a DNS server for the clients:
   1) Current system resolvers
   2) Google
   3) 1.1.1.1
   4) OpenDNS
   5) Quad9
   6) AdGuard
DNS server [1]: 2

Enter a name for the first client:
Name [client]: client
```

Figure 4. 6 Configuring OpenVPN

3.4.6 Setting up Email Server

To set up our email server we are using iRedMail which is a shell script that installs and configures all the necessary mail components on DatumNode which include: Postfix SMPT server, Dovecot IMAP server, Nginx, MySQL, etc.

Step – 1 We purchased a Domain Name "datum-node.com" from pk.godaddy.com

Step – 2 We created several DNS records such as MX, A, CNAME, and SPF. MX record specifies which hosts handle emails for a particular domain name in this case it is "datum-node.com". A Record specifies the IP address of the domain which is datum-node.com. CNAME specifies redirects from one domain to another. SPF (Sender Policy Network) is added in a form of TXT that stores information relation to datum-node.com and it used as an attempt to control forged emails.

Step – 3 We configured the hostname by setting up FQDN (Fully Qualified Domain Name) on our server.

hostnamectl set-hostname mail.datum-node.com

Step – 4 We downloaded the latest version of iRedMail from github.

git clone https://github.com/iredmail/iRedMail.git

Step – 5 Ran the bash script and specified various things which included

- Directory to store user mailboxes - */var/vmail*
- Web server – *Nginx*, for Admin panel and webmail
- Database for backend storage of email accounts – *MariaDB*
- Specified our mail domain – *datum-node.com* and password for the mail domain administrator
- Some additional components; *Roundcubemail* which is webmail built with PHP and AJAX, *netdata* to monitor the system, *iRedAdmin* is used for web-based admin panel and *Fail2ban* that bans the IP address with too many password failures

Step – 6 We can access the webmail.

mail.datum-node.com

Step – 7 We can manage the users by going to the admin panel.

mail.datum-node.com/iredadmin

3.4.8 Setting up Flask API

We use Flask API as a backend to our android application. The API will receive requests from our app and give various responses. The responses vary from storing data to DB, to encryption, hashing and authenticating. [14]

We set different routes for different functions performed by our app, like move from one window to another or to pick an image. The following routes we use are –

/login – to login and authenticate users

/signup – to register a new user

/profile – get profile data for signed in user

/image – upload an image to server

/getimage – download image from server

We use Flask Sqlalchemy to manage our database, store hashes and get user data (images). By default, Flask API listens at port 5000, so this is where we are going to receive our requests. We can make this port safer by using Mac filtering.

3.5 Mac filtering

We use an ‘IP tables’ module on our DatumNode server to filter packets according to mac addresses. This will allow only the DatumNode clients to connect with the server and block all packets to and from other mac addresses.

We use CLI commands such as

Iptables –I INPUT –p tcp –port 22 –m mac ! –mac-source (Client's mac) –j REJECT

This would allow only the client to connect from his desktop, as we are using sftp which uses port 22 and is tcp connection.

To do the same for our android users, we can just change the port to 5000, which is Flask API's default listening port.

CHAPTER 4: METHODOLOGY

4: Methodology

A brief description about the architectures and methods we have used to set up ‘DatumNode’. This includes the basic structure of backend software and their working. Furthermore, it includes all the starting steps needed by users such as registering, to all the possible features. Also, how encryption and hashing is implemented in our code including new DB entries. Furthermore, it shows us how each user can use our personalize VPN and email services.

4.1 Architecture of DatumNode

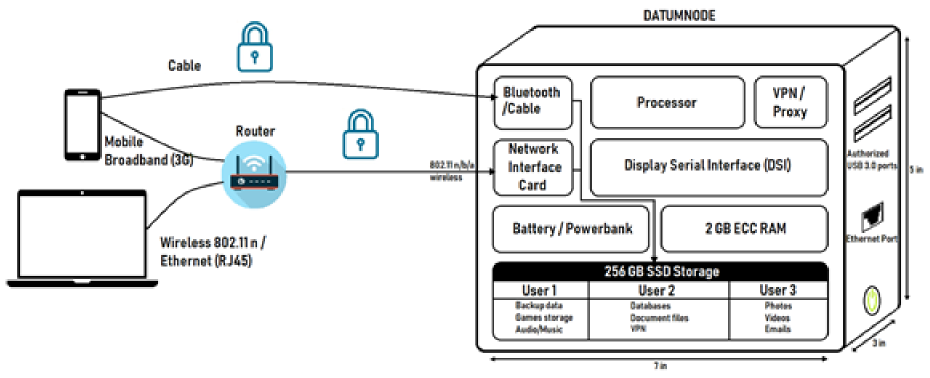


Figure 4. 7 Architecture of DatumNode

DatumNode can work on both PCs and Android phones. Once connected to internet, your device will sync all data (photos, docs, media files and other useful data) to DatumNode. It comes with a built-in email server which can provide free email services to its users, a VPN for encrypted communications, a chat application for all users and encrypted storage of data for all users with full privacy. The product can be customized and comes with a dedicated hardware with good storage and processing.

4.2 Application overview

4.2.1 Desktop Application overview

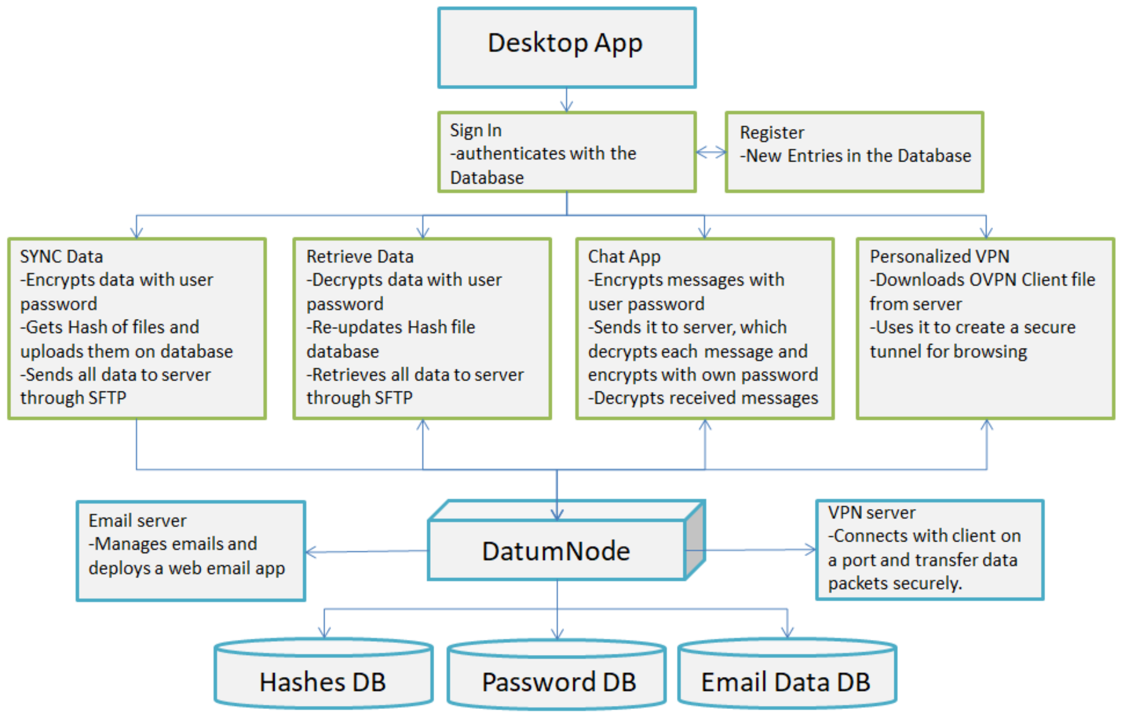

Desktop App
Sign In
-authenticates with the Database
Register
-New Entries in the Database
SYNC Data
-Encrypts data with user password
-Gets Hash of files and uploads them on database
-Sends all data to server through SFTP
Retrieve Data
-Decrypts data with user password
-Re-updates Hash file database
-Retrieves all data to server through SFTP
Chat App
-Encrypts messages with user password
-Sends it to server, which decrypts each message and encrypts with own password
-Decrypts received messages
Personalized VPN
-Downloads OVPN Client file from server
-Uses it to create a secure tunnel for browsing
Email server
-Manages emails and deploys a web email app
DatumNode
VPN server
-Connects with client on a port and transfer data packets securely.
Hashes DB
Password DB
Email Data DB

Figure 4. 8 Desktop App overview

4.2.2 Android Application overview

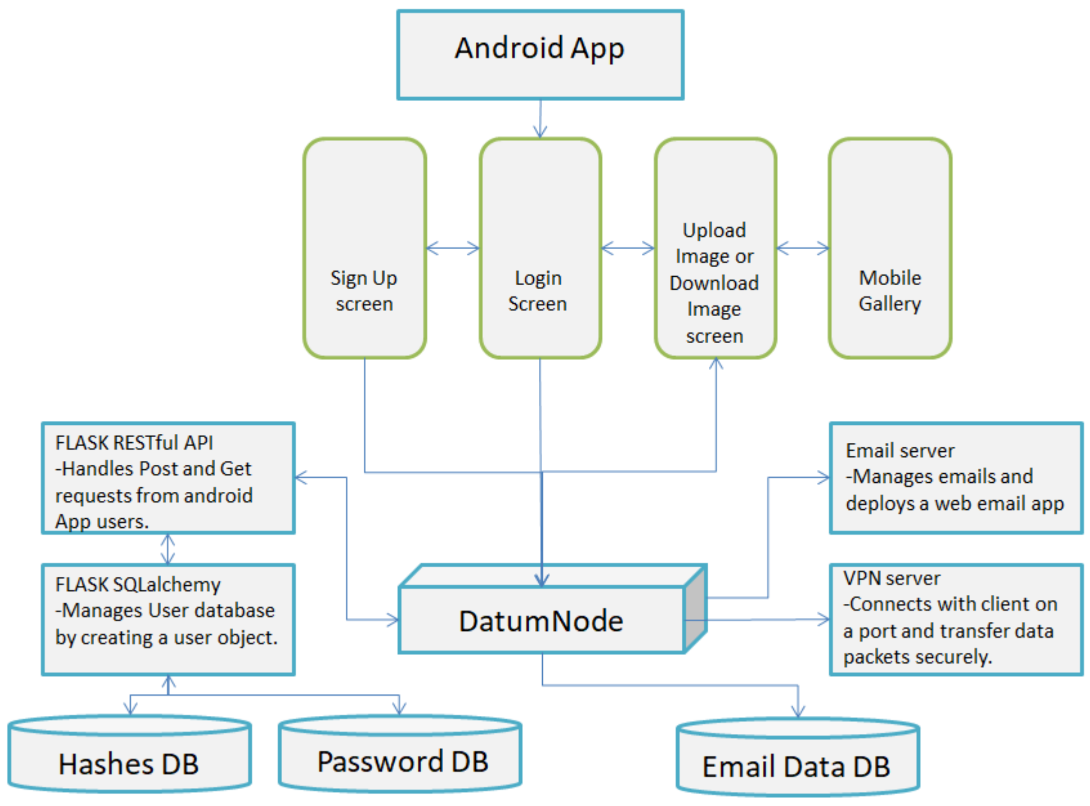

Android App
Sign Up screen
Login Screen
Upload Image or Download Image screen
Mobile Gallery
FLASK RESTful API -Handles Post and Get requests from android App users.
FLASK SQLalchemy -Manages User database by creating a user object.
DatumNode
Email server -Manages emails and deploys a web email app
VPN server -Connects with client on a port and transfer data packets securely.
Hashes DB
Password DB
Email Data DB

Figure 4. 9 Android App overview

4.3 Registration on Desktop

This window is designed in PyQt5 using Qtdesigner. The file is first change from .UI format to .PY. This window is 'called' from the login page window when a new user tries to register.The registration process includes the admin of 'DatumNode' to type in his own password register a particular user. When a user is succesfully registered, he/she will have their data inserted into a database. The databse is then replaced with the one already on the server. This database provides us with usernames for DatumNode's secure chat app and passwords for encryption. Additionally, once registered a user will have his/her very own directory created for them in DatumNode server. Furthermore, only that user will be allowed to access or sync files in the directory. Not even the admin would be permitted to access or make any changes to a user's directory. Usernames can be only used once and passwords cannot be changed .

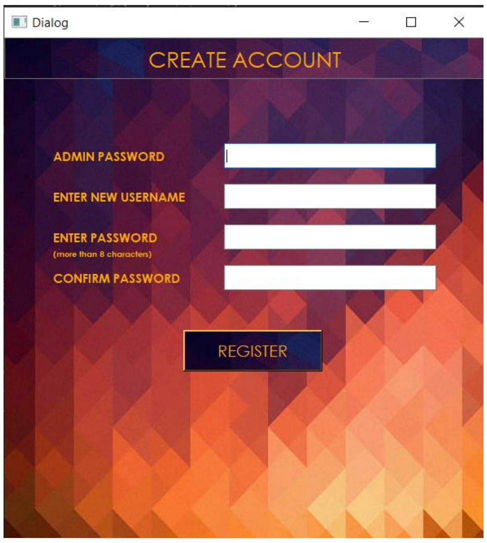


Figure 4. 10 Dialog box

4.3.1 Registration on Android

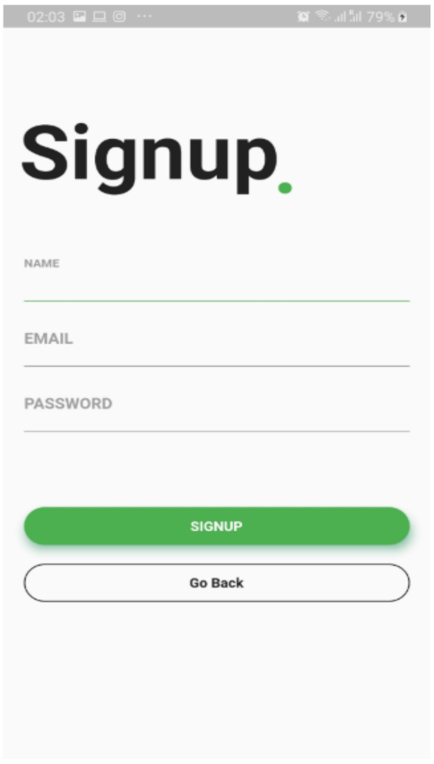
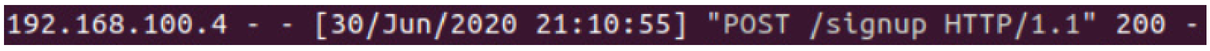


Figure 4. 11 Android Registration Window

These windows are designed in Android Studio using Flutter, Dart. The blackened API used is Flask Restful API on Python.

To register we simply move select the register option in our main window and we are sent to the signup window where we are required to type in our data. Since each client already has their email address assigned to them, they can sign in using that. A user will not be allowed to create an account if he isn't a DatumNode account holder. When a user clicks the Signup button, it generates a POST request to ***new http.Uri("server Ipaddress :port", "/signup")*** and includes the 3 user inputs into body.

The three inputs are then received by FlaskAPI and entered into a database using Flask SQLAlchemy



```
192.168.100.4 - - [30/Jun/2020 21:10:55] "POST /signup HTTP/1.1" 200 -
```

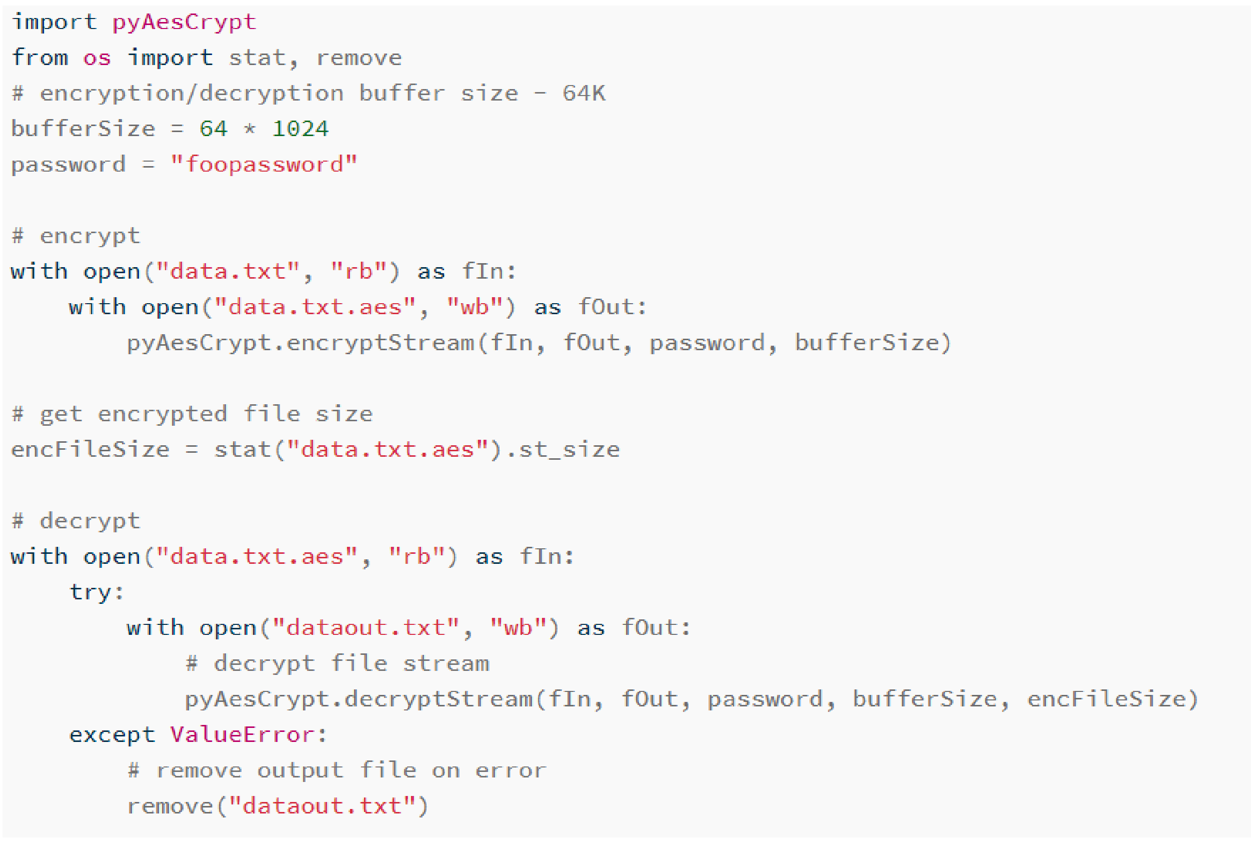
4.4 Encryption

We are using PyAesCrypt in python for our desktop version. This library returns files and binaries in an encrypted form. For our project we will encrypt data using AES 256 CBC.

How it works [12]-

- We Import the library into python
- Define a variable buffer size (64 * 1024 normally assigned)
- Define a password or key to encrypt/decrypt the data with. Normally, we are just using the password a client used to log into DatumNode.
- Then we call onto the PyAesCrypt object, utilizing 'encryptFile' function and passing 'filename', 'New filename', password and the buffer size as arguments and vice versa for decryption.

Pseudocode:

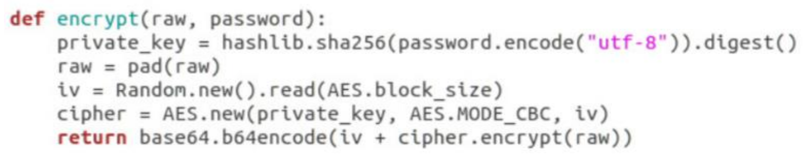


```python
import pyAesCrypt
from os import stat, remove
# encryption/decryption buffer size - 64K
bufferSize = 64 * 1024
password = "foopassword"

# encrypt
with open("data.txt", "rb") as fIn:
    with open("data.txt.aes", "wb") as fOut:
        pyAesCrypt.encryptStream(fIn, fOut, password, bufferSize)

# get encrypted file size
encFileSize = stat("data.txt.aes").st_size

# decrypt
with open("data.txt.aes", "rb") as fIn:
    try:
        with open("dataout.txt", "wb") as fOut:
            # decrypt file stream
            pyAesCrypt.decryptStream(fIn, fOut, password, bufferSize, encFileSize)
    except ValueError:
        # remove output file on error
        remove("dataout.txt")
```

Figure 4. 12 Encryption Pseudo code

For our syncing part, we only use encryption on our client side. DatumNode isn’t allowed or given the ability to decrypt it. However, for our chat application we encrypt binary streams on both our client and server. Thus, keys for our chat application will be different from the syncing ones

Encryption in Flask API



```python
def encrypt(raw, password):
    private_key = hashlib.sha256(password.encode("utf-8")).digest()
    raw = pad(raw)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(raw))
```

Figure 4. 13 Encryption in Flask API

We use AES CBC encryption and encrypt it with the user password, just like the desktop application.

4.5 Hashing

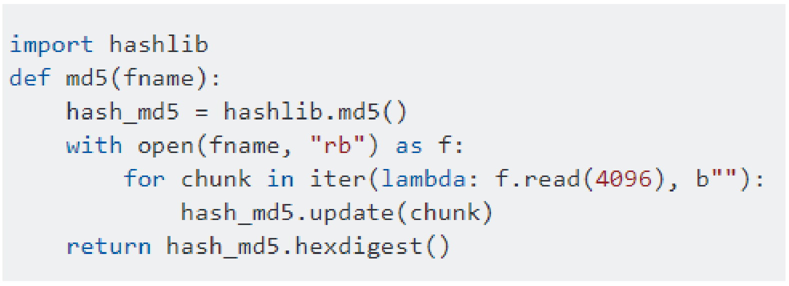
We are using Hashlib in python for our desktop version. This library returns hashes for files before being synced. Hashlib supports a lot of hashing algorithms like SHA256, SHA512, blake2b and md5. [13]

We use md5 as our hashing algorithm for the project as it's much simpler to use. All the hash values of files synced are to be stored on a database. That database is uploaded on DatumNode server when syncing is complete. When the client wants to sync again, the application will 'get' the database from server and compare it hashes to the new files being uploaded. Repeated files will then be discarded. Furthermore, each client will have its own database for hashvalues.

How it works-

- We import the library into python
- Create a hashlib md5 object
- From the object, we call the 'update' method which takes in defined number of binaries from a file as an argument.
- We can then get a 'hex string' in return using 'hexdigest' method or a normal string using 'digest' method.
- These values are uploaded on the database.

Pseudocode

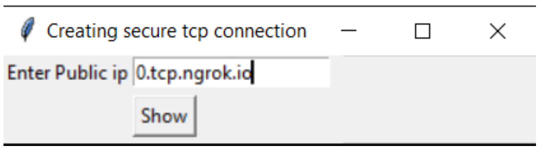


```
import hashlib
def md5(fname):
    hash_md5 = hashlib.md5()
    with open(fname, "rb") as f:
        for chunk in iter(lambda: f.read(4096), b""):
            hash_md5.update(chunk)
    return hash_md5.hexdigest()
```

Figure 4. 14 Hashing Pseudo code

4.6 Desktop App Walkthrough

Step 1 – Run the DatumNode.Exe file in your desktop

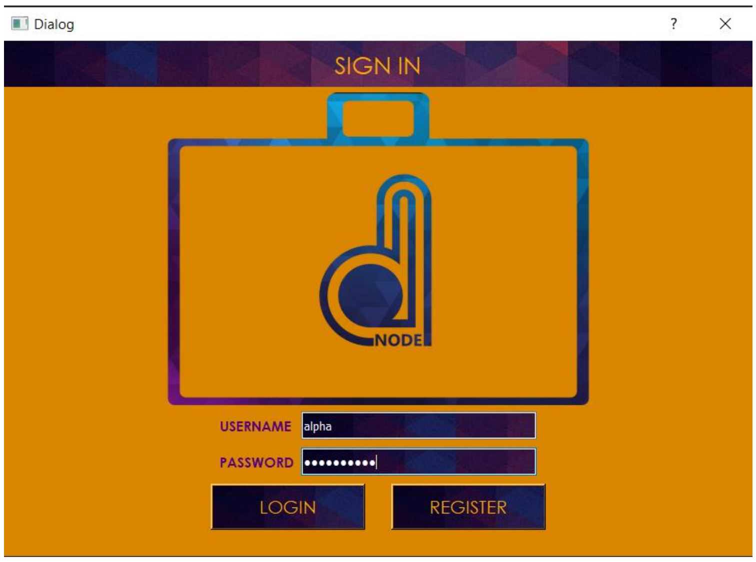


Enter Public ip 0.tcp.ngrok.io

Figure 4. 15 Creating secure TCP connection

When we run the application, a small window pops up. It will require a public/local IP and port to connect with DatumNode.

Step 2 – When we secure a connection, a login window pops up that requires users to sign in using a username and password. Also, new users can be registered by admin (4.3).



USERNAME alpha
PASSWORD ••••••••••

Figure 4. 16 DatumNode login page

Step 3 – After users successfully login, the main window pops up allowing users to choose the following data node features. They can choose to sync data or retrieve it from DatumNode. Moreover, they can choose DatumNode’s secure chat (4.4) and also browse safely through the VPN feature (3.4.5).

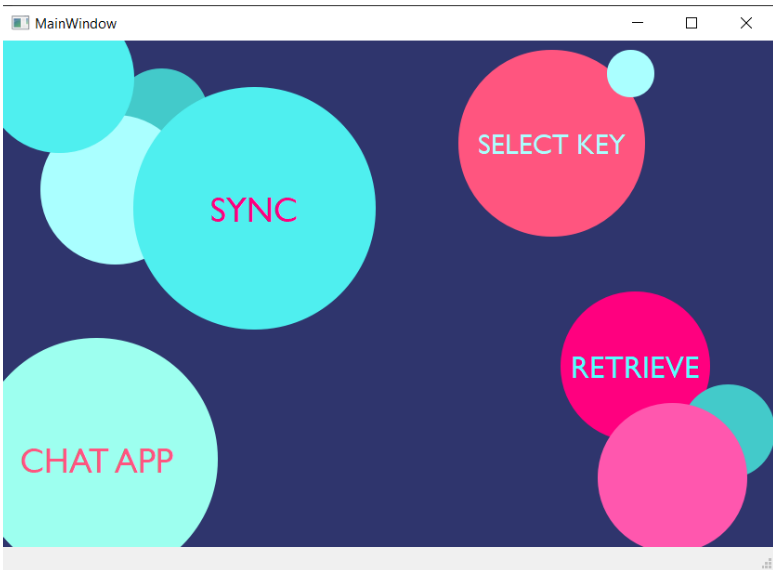


Figure 4. 17 DatumNode UI

Step 4 – When users choose the syncing option DatumNode checks' if the user has its own directory in the server or not. If not DatumNode creates a directory named after the user. After this prerequisite step, a window pops up on the desktop client asking users to choose a folder they want to sync. When a user chooses a folder, our software syncs all the data to DatumNode. The sinking procedure has two important parts, one is the encryption and the other is hashing. Each file that is being synced has its hash value stored in a database to avoid repetitions. Furthermore, each file is encrypted using AES 256 before being synced.

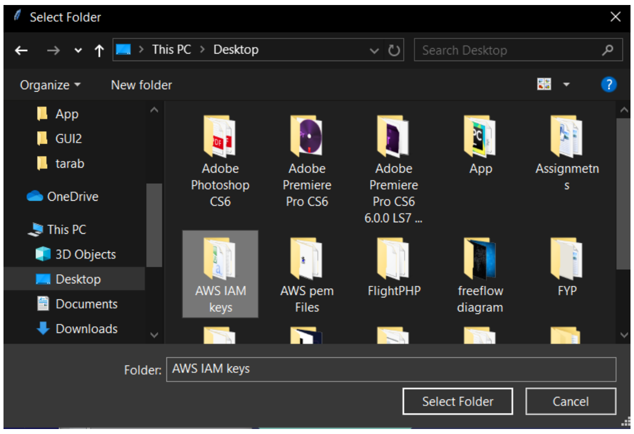


Figure 4. 18 DatumNode sync window

Step 5 – Whereas if users were to retrieve the files, they would simply just click on the retrieve button. A new window pops up listing all the folders present in the users' respective directory. The User can simply choose to retrieve the whole folder or just a file from it. When the file is retrieved it is decrypted and has its hash removed from the database

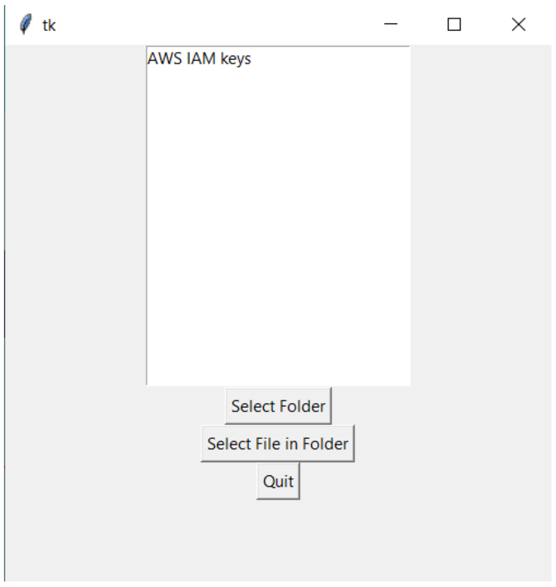
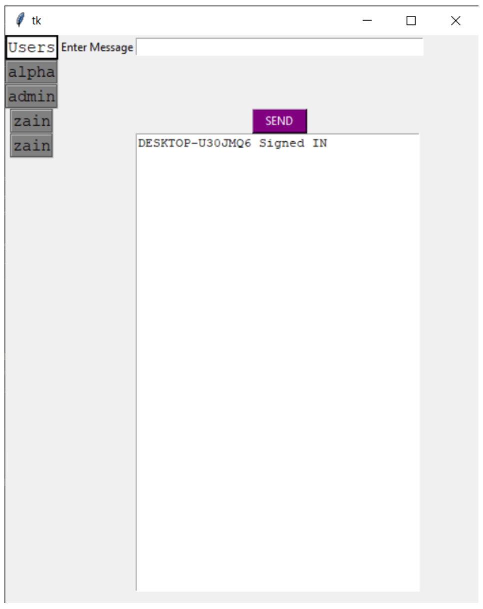


Figure 4. 19 DatumNode retrieval window

Group Chat Application on Desktop



```
DESKTOP-U30JMQ6 Signed IN
```

Figure 4. 20 Client's chat application window

A simple group chat app based on python socket programming. Once a client clicks on the chat button, it's IP address would be appended to a list of connections the server is broadcasting messages to. Each client will send their message packets to the DatumNode server encrypted with their own passwords. DatumNode will receive the packet from a user, match it from the database and decrypt it using the user's password. Then DatumNode will encrypt the packet with its own password (public key) which is known to every client. The message is then broadcasted to all the clients connected at that moment. If a client wants to disconnect, he just has to type in quit. The 'quit' message won't be displayed but the client will be removed from the broadcast list.

For the chat app simply declare a socket on both DatumNode and client application using [9]

s = socket(socket(AF_INET, SOCK_DGRAM)

To have Full Duplex communication, we use threading to continuously receive messages in the background.

Recv = threading.Thread(target = func, args =(Ipaddress,))

Recv.daemon = True

Recv.start()

4.7 Android App Walkthrough

Step – 1

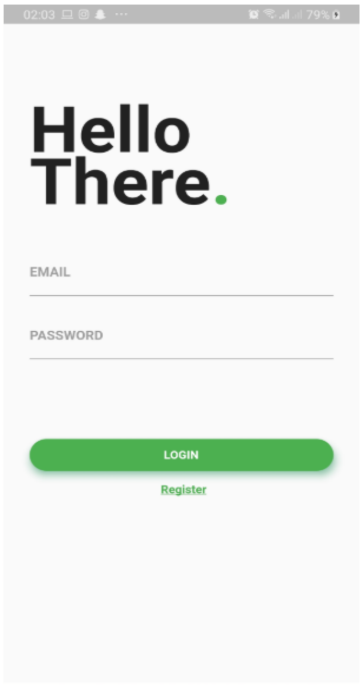


Figure 4. 21 Android Login Window

When we receive a user id and password through our API, we check the password hash with the stored hash in DB. With a success response we move to our next window.

Step – 2

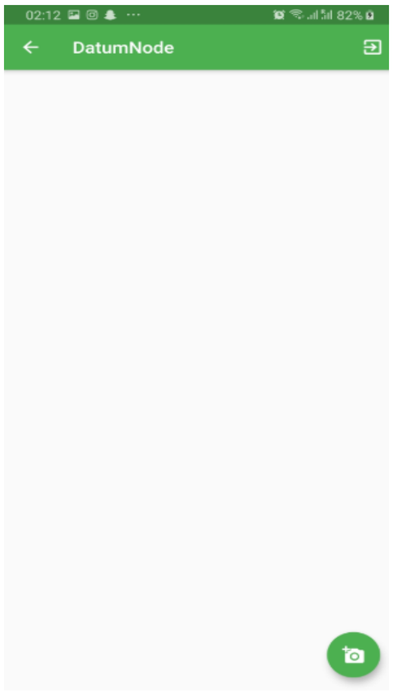


Figure 4. 22 Android Uploaded images Window

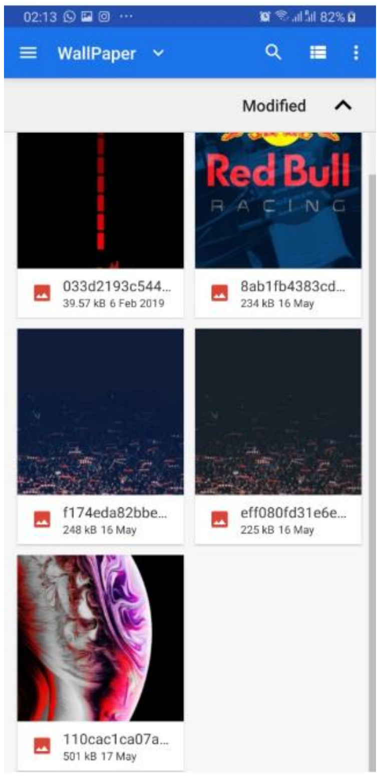


Figure 4. 23 Android Gallery to choose files [15]

To choose an image we touch a button on the bottom right screen. This opens up our mobile gallery and we can choose the image/images we want to upload. The image is then encoded in base64 format and sent to the API in a body as a post request. The body will also include the email for used identification and filename.

Flask Backend

The image received is encrypted and stored in the DB linked with the email.

Step – 3

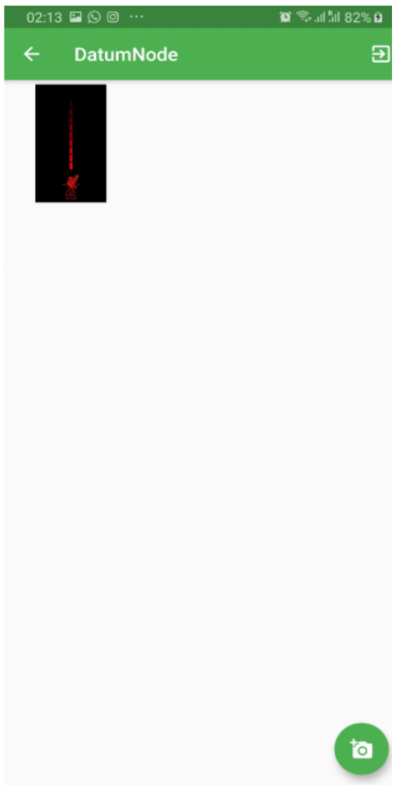


Figure 4. 24 Android Uploaded images window with log out button

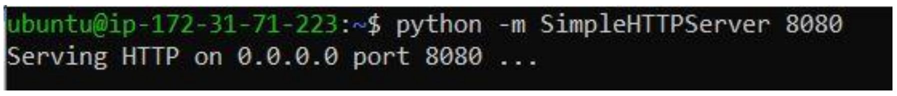
Using a ‘for loop’ we are able to see all the images available for us to retrieve. When we choose a file, a request is generated.

To logout we have to click on the top right icon.

A query is run to show all the images of the user logged in and then returned as jsonify objects.

4.8 Using OVPN on the client side

Step – 1 Client profile is on the server side; to make it available for the client we are using HTTP. We can also use SSH to transfer the client profile to the client.
Making "client profile (client.ovpn) available to the client by HTTP.

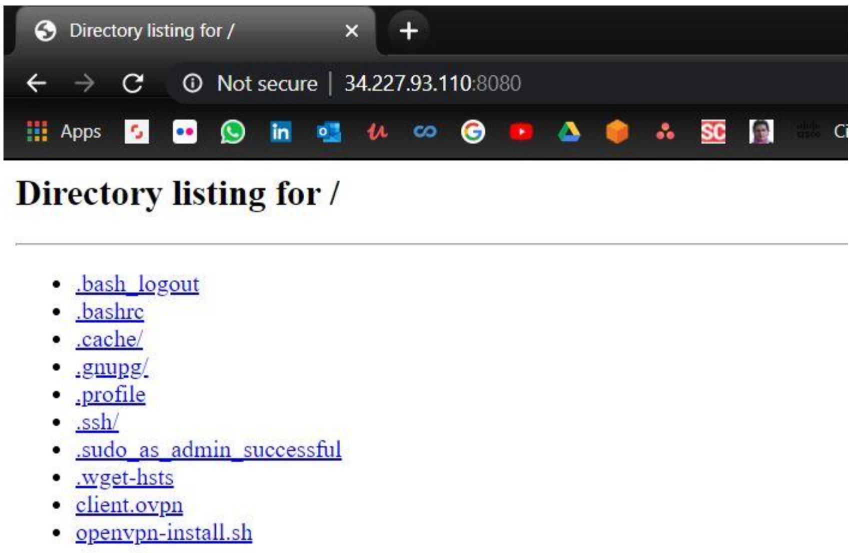


```
ubuntu@ip-172-31-71-223:~$ python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
```

Figure 4. 25 Making client.ovpn available to the client

The steps below are on the Client-side [11]

Step – 1 Downloading "client profile (client.ovpn)"

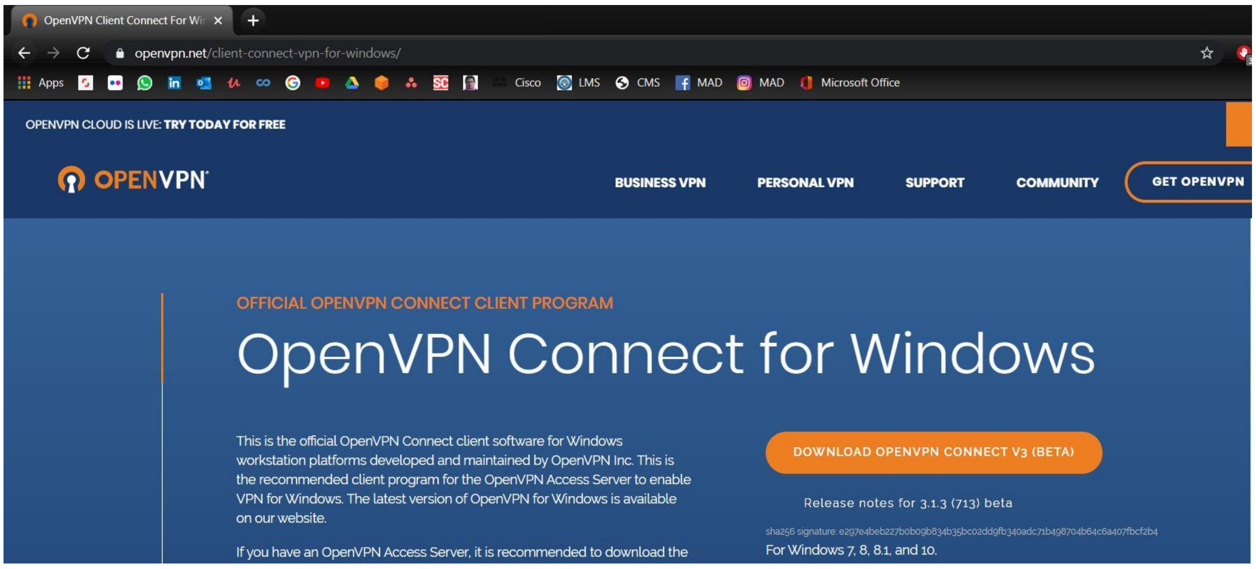


Directory listing for /

- .bash_logout
- .bashrc
- .cache/
- .gnupg/
- .profile
- .ssh/
- .sudo_as_admin_successful
- .wget-hsts
- client.ovpn
- openvpn-install.sh

Figure 4. 26 Downloading client.ovpn

Step – 2 Downloading OpenVPN Connect on the client side. OpenVPN Connect is used to access the OpenVPN server which is DatumNode.


OpenVPN Client Connect For Wi
openvpn.net/client-connect-vpn-for-windows/
OPENVPN CLOUD IS LIVE: TRY TODAY FOR FREE
OPENVPN
BUSINESS VPN
PERSONAL VPN
SUPPORT
COMMUNITY
GET OPENVPN
OFFICIAL OPENVPN CONNECT CLIENT PROGRAM
OpenVPN Connect for Windows
DOWNLOAD OPENVPN CONNECT V3 (BETA)
Release notes for 3.1.3 (713) beta
For Windows 7, 8, 8.1, and 10.

This is the official OpenVPN Connect client software for Windows workstation platforms developed and maintained by OpenVPN Inc. This is the recommended client program for the OpenVPN Access Server to enable VPN for Windows. The latest version of OpenVPN for Windows is available on our website.

If you have an OpenVPN Access Server, it is recommended to download the

Figure 4. 27 Downloading OpenVPN

Step – 3 We imported the client profile (client.ovpn) on OpenVPN Connect which we downloaded through HTTP. By importing the client profile, the user will be able to access the OpenVPN server. .

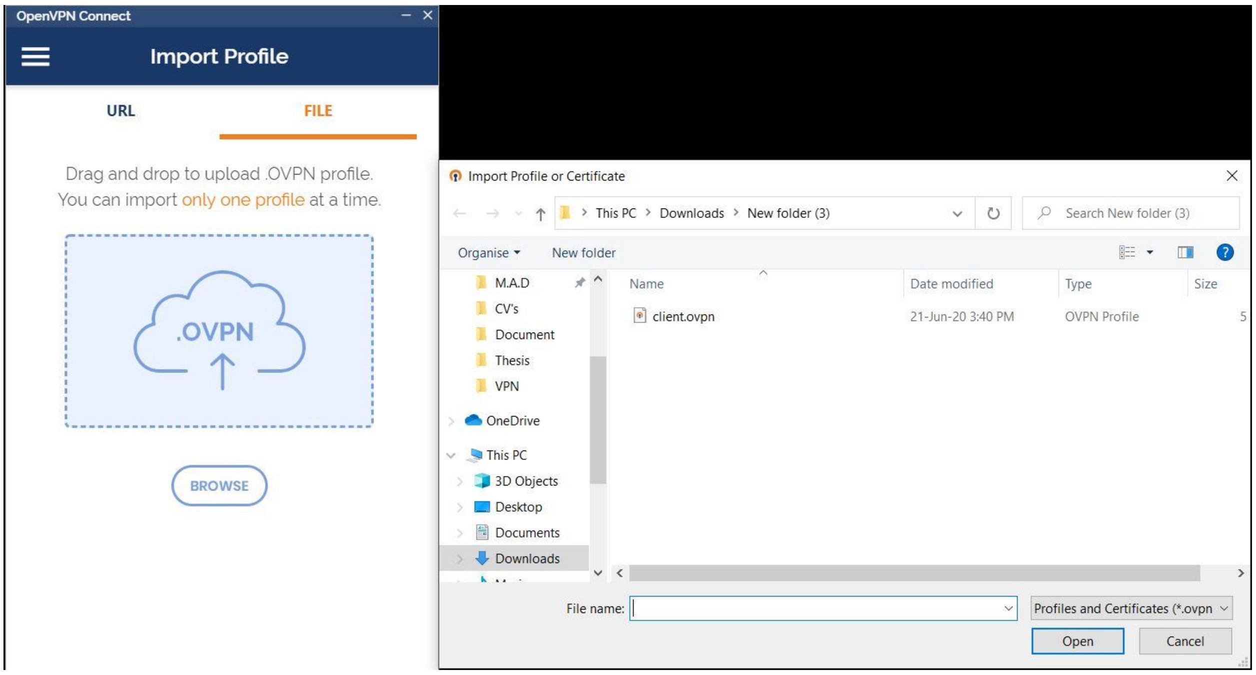


Figure 4. 28 Importing client.ovpn on OpenVPN Connect

Step – 4 Connection established. Once successfully connected, we can confirm it by checking our IP address.

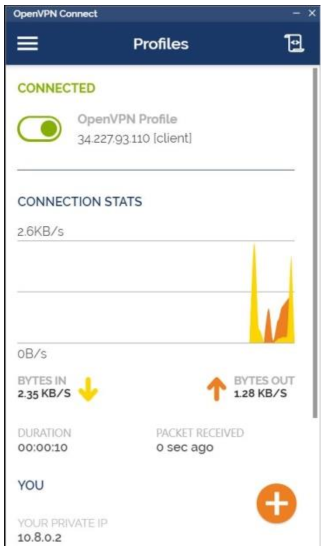


Figure 4. 29 Connection established

4.9 Sending & Receiving Emails

To send and check our email inbox, we search for **URL:datum-node.com/mail**. Here we would be provided with a Login Screen.

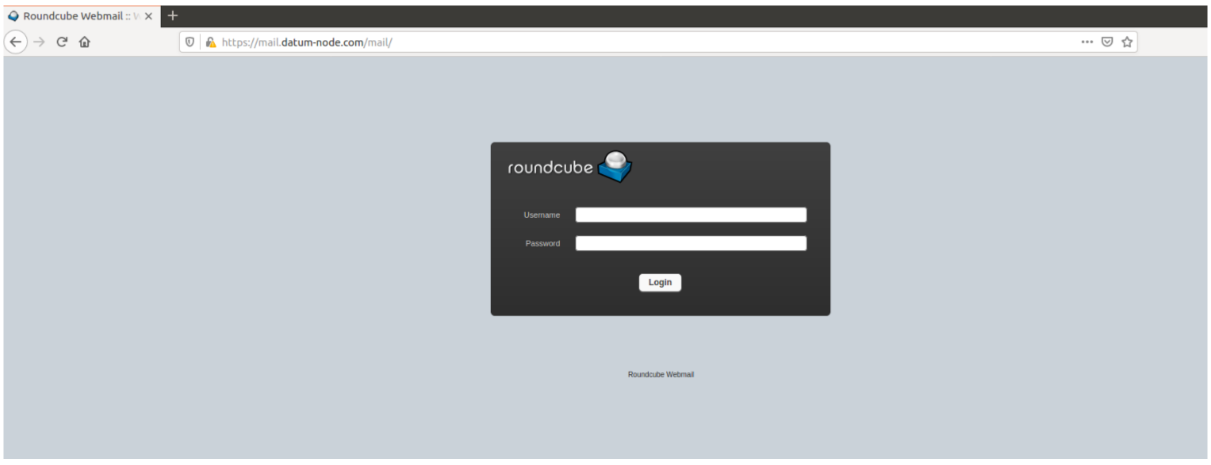
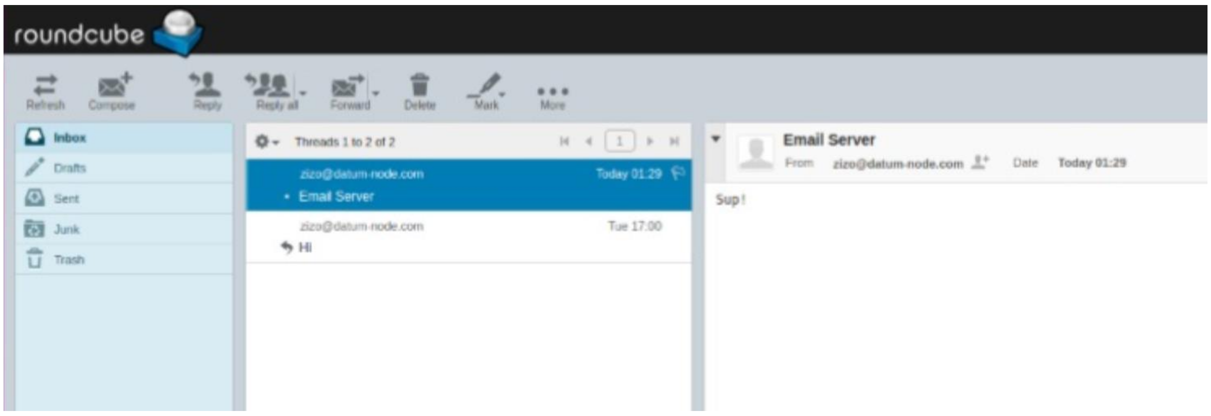


Figure 4. 30 DatumNode mail login screen

After we enter our credentials we would be sent to the main screen where we can send emails and view or inbox.



zizo@datum-node.com Today 01:29
Email Server

zizo@datum-node.com Tue 17:00
Hi

Email Server
From zizo@datum-node.com Date Today 01:29

Sup!

Figure 4. 31 DatumNode mail inbox

CHAPTER 5: RESULTS & ANALYSIS

5: Results & Analysis

A Project block diagram for our to get a better idea and analysis review on how things are set on the client side and how things are working on the backend as well.

5.1 Project Block Diagram

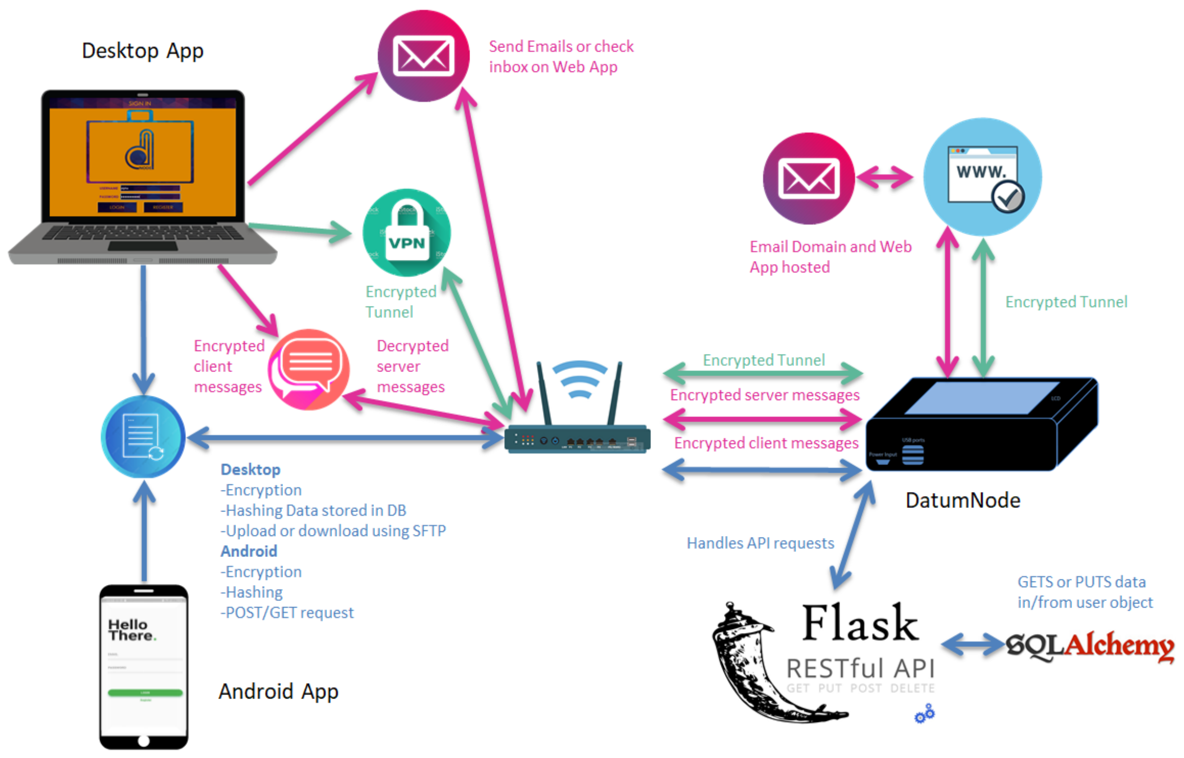

Desktop App
Send Emails or check inbox on Web App
Encrypted Tunnel
VPN
Encrypted client messages
Decrypted server messages
Desktop
-Encryption
-Hashing Data stored in DB
-Upload or download using SFTP
Android
-Encryption
-Hashing
-POST/GET request
Hello There.
Android App
Email Domain and Web App hosted
www.
Encrypted Tunnel
Encrypted Tunnel
Encrypted server messages
Encrypted client messages
DatumNode
Handles API requests
Flask
RESTful API
GET PUT POST DELETE
GETS or PUTS data in/from user object
SQLAlchemy

Figure 4. 32 Block diagram

5.2 Initiation

The device just has to be turned on. The SSH server and Flask API will start automatically. The Email domain will also go online. The user would just have to connect the device to a Wi-Fi and DatumNode will do the rest!

5.3 Objectives

The objectives of 'DatumNode' are:

- To come up with a personalized hardware-based solution for secure data storage and retrieval.
- Windows and Android UI’s to ease client’s access to all the services provided by DatumNode.
- Protect the sensitive contents of your inbox and communicate privately by email using a custom domain.
- A VPN to ensure secure browsing.
- A secure chat app for unified communication.
- DatumNode server to be as secure as possible with reliable authentication to access.
- This would open further avenues of research in the security and privacy domain as it can be further enhanced to make the product commercially viable.

The device just has to be turned on. The SSH server and Flask API will start automatically. The Email domain will also go online. The user would just have to connect the device to a Wi-Fi and DatumNode will do the rest!

CHAPTER 6: CONCLUSION & FUTURE WORK

6: Conclusion & Future work

A brief summary on what DatumNode is again and how is it easing our daily lives. However, DatumNode is still not perfect. That being said, here are some ideas that would bring it closer to perfection.

6.1 Conclusion

In conclusion, DatumNode is Pakistan's first indigenous solution for Android phones and Windows for seamless storage and sharing of data, along with a unified communication platform for its customers. Taking into consideration that businesses spend millions on storage providers in order to securely and privately store their data. Therefore, DatumNode can very easily be adopted as surveillance issues have accentuated the need to have indigenous, handy, well-controlled storage solutions that provide data privacy and confidentiality. With the rising data breaches and identity thefts, our solution gives you the complete ownership of your data as it is pocketable and comes with extendable storage.

DatumNode is responsible for all these features-

- Encrypting all kinds of data on the device including databases
- Providing secure connection to clients for file sharing/syncing
- Securing all kinds of online activity though our hosted Virtual Private Network
- Hosting a chat server that decrypts all incoming messages and encrypts outgoing messages to clients
- Lastly, a secure local email domain hosted on the device which can be made public using NGROK

6.2 Future Work

There are enhancements that could make DatumNode even more user friendly and give admins much more control on what is going around without victimizing a user's privacy!

6.2.1 Software Enhancements

At this moment DatumNode has a directory for each user. Only that user can edit or delete the directory. However, all DatumNode requires is a password. So, we can add Multi Factor Authorization for an extra layer of security. [16]

Secondly, we could have a shared folder accessible by multiple users set by an admin. For example, setting up granular permissions for a directory would help organizations and departments greatly.

6.2.2 Hardware Enhancements

Efficiency and latency should be key while accessing DatumNode. As long as DatumNode is responsible for hosting multiple services, it could end up being slow provided not being given sufficient RAM. Also, extra lines of code could make a program slower. Furthermore, if multiple users were to access DatumNode all at once, it could entirely slow down the process. So, in the end having a fast, convenient and unbreakable connection should be what we aim for in the future.

That said, DatumNode is also vulnerable to DOS attacks when made public. Some router and hardware firewalls can help prevent that. They would also help detect network intrusion and provide a much safer and secure environment. [17]

References

[1] C. S. Statistics, 2019. [Online]. Available: https://www.thesslstore.com/blog/80-eye-opening-cyber-security-statistics-for-2019/.

[2] [Online]. Available: https://www.thehelm.com/.

[3] [Online]. Available: https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview.

[4] [Online]. Available: https://pypi.org/project/PyQt5/.

[5] [Online]. Available: https://www.ovpn.com/en/guides/ubuntu-cli.

[6] [Online]. Available: https://developer.android.com/studio/install.

[7] [Online]. Available: https://flutter.dev/docs/get-started/editor.

[8] [Online]. Available: https://ngrok.com/download.

[9] P. C. Application. [Online]. Available: https://github.com/anuragdwivedy/Insecure-UDP-Python-Chat-Server-Client.

[10] [Online]. Available: https://openvpn.net/community-resources/how-to/.

[11] [Online]. Available: https://www.ovpn.com/en/guides/windows-openvpn-gui.

[12] [Online]. Available: https://pypi.org/project/pyAesCrypt/.

[13] [Online]. Available: https://docs.python.org/3/library/hashlib.html.

[14] [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/api/.

[15] [Online]. Available: https://flask-sqlalchemy.palletsprojects.com/en/2.x/.

[16] MFA. [Online]. Available: https://auth0.com/docs/mfa.

[17] C. R. 1800. [Online]. Available:

https://www.cisco.com/c/en/us/td/docs/routers/access/1800/1801/software/configuration/guide/scg/firewall.html.

[18] [Online]. Available: https://www.linuxbabe.com/mail-server/ubuntu-18-04-iredmail-email-server.

Appendix A: Synopsis Form – DatumNode

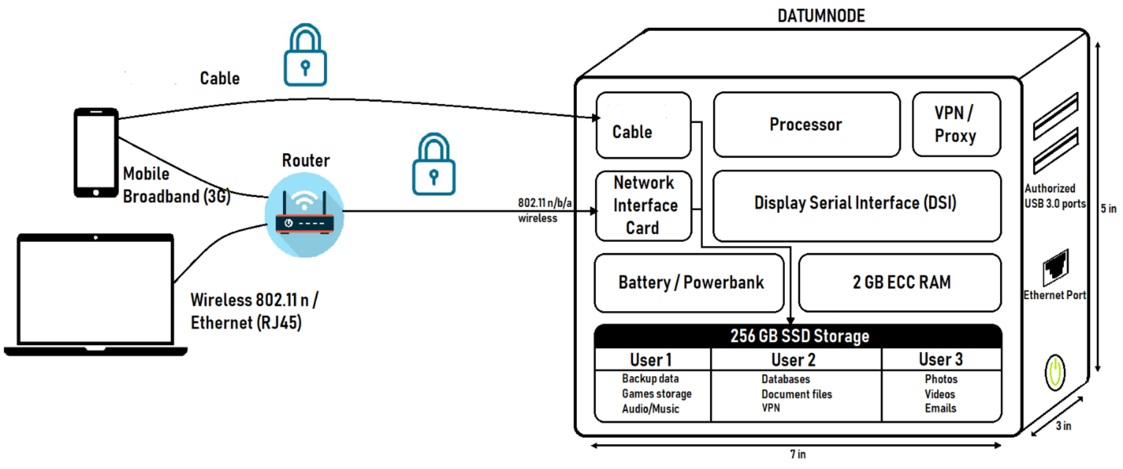
Extended Title: A personalized secure data repository

Brief Description of The Project / Thesis with Salient Specifications:

The ubiquity of internet and mobile services has produced enormous amount of data, not just for corporates but for individuals as well. This mammoth use of internet is basically triggered by the use of smartphones. Nearly 3 Billion people use smartphones [1] across the globe while 21 percent of Pakistan's nearly 220 million population use the mobile internet services [2]. This phenomenon gave rise to a lot of storage issues at the user end thereby giving rise to the cloud technology whereby data and services are hosted on the cloud. A lot of service providers offer storage and processing services and now the entire digital life (photos, emails, calendar, files, videos etc.) of an Android user resides on the Google servers. In this case, user has complete access and control over his data but he is not the only one having that control thereby breeding a bunch of trust issues for users who are privacy conscious and do not want these big companies to spy on them.

'DatumNode' provides a solution to the problem for all privacy conscious people and smaller organizations. It protects files, photos, media, contacts and other important data by storing it on a device of your own. It would also offer a personalized email domain and can act as a VPN to further protect users' online activity. 'DatumNode' would basically replace these free storage services so that only the user will have access and complete control over his data. 'DatumNode' can also support limited number of multiple users in order to make it useful for smaller organizations. It would be a hardware device with large storage and processing capacity along with a network interface that can be accessed over the internet or via data cable (in case of device proximity).

Diagram



Scope of Work:

To develop a secure personal server to securely store and protect your personal information so that only the user has access to it. The overall scope of this project includes, but is not limited to:

- Computer Networks (TCP/IP Architecture)
- Network Security (Encryption, VPNs)
- Programming (Sockets, Web, Android, File System)
- Hardware (Interfacing, Cable and WiFi connectivity, SSD)
- Operating System (Windows, Android, Ubuntu)

Academic Objectives:

- To come up with a personalized solution for secure data storage and retrieval that would also act as an email server and VPN.
- This would open further avenues of research in the security and privacy domain as it can be further enhanced to make the product commercially viable.

Application / End Goal Objectives:

This hardware device can be placed anywhere the user wants let it be his home or workplace. The server is accessible via Wifi, allowing the user to sync his data from anyplace anytime as long as he has access to the internet

Previous Work Done on The Subject:-

- **Tonido** server allows you to access all your files on your computer from a web browser, smartphone, tablet or even DLNA (Digital Living Network Alliance) enabled devices. However, it is mainly for people that invest their data on a single computer rather than other devices.

- **HELM** is a personal server that lets you break from cloud and safely store your files and photos in it. Protects sensitive contents of your inbox, sends and receives private and secure emails from your custom domain. However, it only has 120 GB of storage space and your data backup is stored in their off-site personal server.

- **Upcloud** provides one of the fastest cloud servers (faster than SSD). But powerful control panel and API lets you spend more time coding and less time managing your cloud server infrastructure.

- **Sync.com** is also one of the other best free cloud storage available. Syn.com provides a free storage of about 5GB. The best feature of the Sync.com is the restoring of deleted files and thereby stops users from regretting for mysteriously lost important documents.

- **pCloud** is one of the best free cloud storage options that provides 2TB storage and its unlimited remote upload traffic feature makes it unique. Quick and easy, stable and great GUI. Its desktop application is slow and the android app is buggy and generates a lot of errors.

Reference links:

[1]. https://venturebeat.com/2018/09/11/newzoo-smartphone-users-will-top-3-billion-in-2018-hit- 3-8-billion-by-2021/

[2]. https://profit.pakistantoday.com.pk/2018/04/21/pakistans-mobile-internet-stand-at-21- percent-of-the-population-report/

Material Resources Required: Raspberry pi, 256 GB SSD, Battery/Powerbank

No of Students Required: 04

Group Members:

- Muhammad Usman (Syndicate leader)
- Shaikh Muhammad Zain
- Ayesha Gillani
- Ahmed Rauf Khan

Special Skills Required:

- Python
- Andriod development
- Linux

Approval Status

Ass. Prof Waleed Bin Shahid

Assigned to: __________________ **HoD Signature** ___________________________

R&D SC Record Status File # ______ **Coordinator Signature** ________________