**INTERNSHIP REPORT**



**Submitted By**:

Shruti Priya

Python Developer intern

CBDNS

**ACKNOWLEDGEMENT**

I want to thank my mentors and advisers and everyone at the company for their patience and assistance during my virtual training. Thanks to their guidance, I was able to develop Python bot.

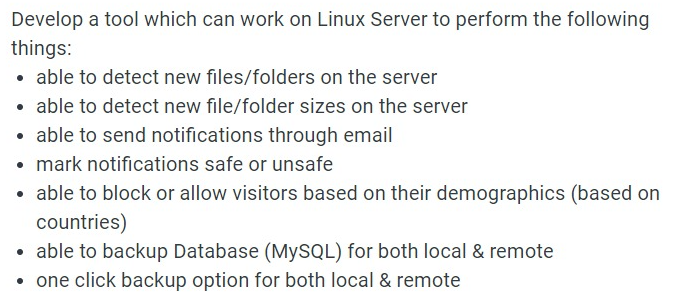
These skills will help me to expand my resume and advance my career.



Shruti Priya

**REPORT**

As a python developer I was assigned the task to create python scripts which would perform the following tasks:



Three different python scripts were developed to achieve these tasks.

***SCRIPT 1: monitor.py***

This python script uses various packages.

External packages:

* watchdog.observers – for observing changes in the file/folder structure
* watchdog.events – for capturing events of change in the file/ folder structure

In-built packages:

* os – for extracting System variables
* time – for scheduling the observer event
* smtplib – for sending mail

Private external file

* file\_types – contains a list of potential file extensions

The script first imports variables from external file *file\_types*, and store the various file extensions. It further declares two variables which stores the email address of the receiver and the absolute path of the folder which needs to be tracked.

The *send\_mail()* function takes in three parameters – email address of the receiver, the subject of the email and the body message of the email. Initally, the username and the password is extracted from the system variable, with the help of *os* library. Then, a smtp connection is established. Finally the function sends the mail to the receiver of the mail specified in the parameters.

The *handler* class inherits from the *FileSystemEventHandler* class from the watchdog package. Two event listeners are defined as functions *on\_created()* and *on\_modified().*  The former function is called whenever a new file or folder is created. Based on the file-type, a mail is sent to the receiver with different urgency, based on the type of file created. The size of the file is also sent in the mail.

The *on\_modified()* function similarly, sends mail with varying levels of urgency, along with the file name and the file size.

Finally an instance of the *Observer* class and *Handler* class is created. The observer is scheduled to observe the given folder. Since recursive is true, all the subfolders of the given folder will also be mentioned.

***SCRIPT 2: demographic\_filtering.py***

The second script is the demographic\_filtering.py. It uses the following packages and third party API:

*External package*: Flask

*Default package*: requests

*External file*: blacklisted\_countries

*Third party API*: ipapi.co

It is a simple flask server which runs of port 5500 of the localhost. The server responses any get request made to the home URL ( / ). It first extracts the IP address of the incoming request. It then calls the function *filter\_by\_country()* which takes in the ip address as the parameter and returns whether the IP address should be served or not in form of true or false.

The main functionality is wrapped in the *filter\_by\_country()*  function. It takes in IP address as the input. The function then calls a third party API provided by ipapi.co. The format of the get request is:

*https://ipapi.co/{IP Address}/{format}*

If the IP address is a Reserved IP address, then it does not contain any country name and the request must be served, hence a True value is returned. Else if the country of the incoming IP address exists, then it is matched with the countries specified as blacklisted. The list of countries blacklisted in saved in a different file called *blacklisted\_countries.* Based on whether a match is found or not, a False and a True value is returned respectively.

***SCRIPT 3:db\_backup.py***

The third script is a one click option to take backup of the database. It uses only two default libraries of python.

* os – for extracting the value of the system variable, and running system command
* ftplib – for transferring file to remote server

Additionally, this script requires that Mysql is installed in the machine, and the path is added to environment variable so that the script can be run from anywhere on the local machine.

The password of the root user is extracted from system variable. The database variable stores the name of the database whose backup is to be taken.

For local backup, the os.system method is used to run the shell command. *mysqldump* is the command line command for taking backup. It is followed by the username, the password, the server and the name of the database. The output is redirected to be stored in the file named as *database\_backup.sql.*

For transferring the backup file to the remote server, FTP is used. The FTP method of the ftplib is used to establish a session to the remote server with the given IP address. The username as well as the user password must also be passed to the FTP method. After transferring the file, the file is promptly closed, followed by closure of the FTP session.

**CONCLUSION**

From my experience at CBDNS, I was able to get a better understanding of how the industry works and how effective it is. The work experiences I encountered during the internship allowed me to develop skills related to web applications and databases. The overall experience was positive, and everything I learned will be useful in my future career in this field.

I found the python developer internship experience to be positive, and I'm sure I will be able to use the skills I learned in my career later.