Automate Checking for New Notices on DTU's Official Website

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR INNOVATIVE WORK
UNDER

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Pankaj Lahoti
(2K20/B5/47)

Prince Mendiratta
(2K20/B5/69)

Under the supervision of:

MS. PRIYA SINGH
(ASSISTANT PROFESSOR, CSE)
Department of CSE, DTU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi – 110042

CANDIDATE'S DECLARATION

We, (Pankaj Lahoti, Prince Mendiratta, 2K20/B5/47, 2K20/B5/69) students of B.TECH (COMPUTER SCIENCE AND ENGINEERING) hereby declare that the Innovative Project Report titled "Automate Checking for New Notices on DTU's Official Website" which is submitted by us to Department of Computer Science and Engineering, Delhi Technological University, Delhi, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship or other similar title or recognition.

Place: Delhi	Pankaj Lahoti (2K20/B5/47)
Date:	Prince Mendiratta (2K20/B5/69)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CERTIFICATE

I, hereby certify that the Project titled “Automate Checking for New Notices on DTU’s Official Website” which is submitted BY Pankaj Lahoti, Prince Mendiratta, (2K20/B5/47, 2K20/B5/69), Department of Computer Science and Engineering, Delhi Technological University, Delhi, as part of Innovative Work is a record of project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: DTU, Delhi

Date:

MS. PRIYA SINGH
(Assistant Professor, CSE, DTU)

ACKNOWLEDGEMENT

We are very thankful to Ms. Priya Singh (Assistant Professor, Computer Science Eng. Dept.) and all the faculty members of the Computer Science Engineering Dept. of DTU. They all provided immense support and guidance for the completion of the project undertaken by us.It is with their supervision that this work came into existence.

We would also like to express my gratitude to the university for providing the laboratories, infrastructure, test facilities and environment which allowed us to work without any obstructions.

We would also like to appreciate the support provided by our lab assistants, seniors and peer group who aided us with all the knowledge they had regarding various topics.

Pankaj Lahoti (2K20/B5/47) ,

Prince Mendiratta (2K20/B5/69)

B.TECH (CSE)

ABSTRACT

Creating an automated bot to check DTU's official website, https://dtu.ac.in, at regular intervals for any new notices and sending a notification in case of a new update. Coded in python, it utilizes various open-source libraries and services to get this task done automatically. As it aims to be checking the website regularly at fixed intervals, it needs to support being hosted and executing on a VPS (Virtual private server) or a cloud service like Heroku so that we need not have our own dedicated device to be running at all times. Through observation, we have noticed that each notice / announcement has a specific HTML tag for date. Using web scraping via the requests module in python, we get the dates of the notices. Using the dates as the key element, we store the dates in a specific file and then scraping the website again after a fixed interval, we cross check the dates and if there has been any changes, we thus send the latest notice on Telegram via Telegram API. The bot program allows other people to use the bot itself without hosting or deploying the bot. We use MongoDB, a document-oriented NoSQL database used for data storage on the cloud, to record the user details of the people subscribed for notifications which is then further used to broadcast the notification upon a new notice being uploaded.

INDEX

ABBREVIATIONS

Abbreviation	Full Form
API	Application programming interface
vars	Environment Variables
VPS	Virtual Private Server
Bot	Telegram bot
Official Website	https://dtu.ac.in

FLOWCHARTS

Web Scraping -

https://whimsical.com/flowchart-request-py-MzWDL9YLMW4FweZLM9HbaY

Broadcast Plugin -

https://whimsical.com/flowchart-broadcast-py-LYPjcpVLRpH84cFyDApvwe

CHAPTER - 1

INTRODUCTION

The technology by which a process, procedure or task is performed with minimal human interference or involvement accomplished with the aid of technological devices or services is termed as Automation. It is the technique of making a process or a system operate automatically to accomplish a task through a set of instructions or policies.

Automation crosses all functions within almost every industry from installation, maintenance, manufacturing, marketing, sales, medicine, design, procurement, management, etc. Automation has revolutionised those areas in which it has been introduced, and there is scarcely an aspect of modern life that has been unaffected by it.

Automation generally allows monotonous and repetitive tasks which have to be completed manually to be done automatically with the use of defining and deploying a set of instructions or procedures and allowing it to be completed without much, if not zero human interference.

1.1 Background and Motivation

Any new notices or announcements are published on DTU's official website, https://dtu.ac.in . It is important to be informed about the current happenings and upcoming events and instruction which have been issued by the college authorities. Hence, it becomes a responsibility to regularly check the website for new notices and announcements. Practically however, it is simply impossible to have a schedule to check the website regularly over a long period of time.

Thus, we thought about how we can automate the process of checking for new notices on the website regularly and simply get a notification or alert whenever a new notice is uploaded on the website. Hence, we've developed a program that can perform this task automatically within a fixed time interval and in case of any updates, get a notification directly to our mobile or desktop gadgets. From this point onwards, we'll refer to this program as bot.

1.2 Objectives and Scope

The objective of this project is to enhance our knowledge and get some experience with automation. Our core purpose is to check DTU official website and upon receiving a new notice, get a notification on our Mobile device or PC of the same. Our further objectives include creating a Database and allowing users who haven't deployed the bot itself can still use the features.

Our project delves into the territory of web scraping via python, using Telegram chat service for sending notifications, delivering messages through bots in Telegram with the help of Telegram API.

The scope of the project includes:

- Seamless web scraping of DTU's official website at a fixed interval.
- Making sure all categories of notices are being checked for at the site.
- Technical Simplicity and ease of access for users.
- Managing Database of users.
- Proper integration of Pyrogram Library with Telegram API

CHAPTER - 2

General Characteristics

2.1 Modes Of Deployment

Since we want the bot to check the website at a regular interval daily, we need to allocate a device or server so that the bot can run the whole time. Hence, we have multiple possible ways to execute the program, some of which are as follows -

- On a Windows PC
- On a Macintosh
- On a GNU/Linux system
- On a VPS
- On a cloud server, like Heroku or Azure
- On an online hosting or web server
- On an Android Device

Depending on the device, the requirements for the project deployment may differ.

2.2 Requirements

2.2.1 Hardware Requirements

- PC / VPS / Mobile according to the mode of deployment.
- Uninterrupted Internet Connection

2.2.2 Software Requirements

- Python 3
- Termux app (on Android)
- Git package

2.2.3 Python Library Dependencies

- Pyrogram
- TgCrypto
- python-dotenv
- lxml
- bs4
- requests
- urllib3
- pymongo
- dnspython

2.2.4 Front End / Back End

- Telegram account for Front End
- VsCode for programming

2.3 Users of the Project

- ADMIN: The one who hosts and deploys the bot. Responsible for supervision of logs and if any bugs found in the logs.

- USERS: The ones who don’t know anything about programming or deployment but are still able to use the Telegram bot and it’s services.

2.4 General Constraints

- Need for a Telegram account.
- Need to install Telegram app for Mobile devices and PC.
- Uninterrupted Internet connection for web scraping.
- A lot of dependencies which may be complicated to understand on first use.
- Need a MongoDB database and it’s connection URL.
- Need for a Heroku account for deployment.

CHAPTER - 3

Core Concepts and Frameworks

3.1 Telegram

Telegram is a freeware, cross-platform, cloud-based instant messaging (IM) software and application service. Various Telegram client apps are available for desktop and mobile platforms including official apps for Android, iOS, Windows, macOS and Linux. There is also an official web interface and numerous unofficial clients that make use of Telegram's protocol. Telegram has over 500 million monthly active users and is one of the 10 most downloaded apps in the world.

3.2 Telegram Bot API

Bots are third-party applications that run inside Telegram. Users can interact with bots by sending them messages, commands and inline requests. Telegram Bots can be controlled using HTTPS requests to Telegram Bot API. Messages, commands and requests sent by users are passed to the software running on our server. Telegram's intermediary server handles all encryption and communication with the Telegram API for us. We communicate with this server via a simple HTTPS-interface that offers a simplified version of the Telegram API. We call that interface the Telegram Bot API.

3.3 Web Scraping

Web scraping is the process of gathering information from the Internet. The term “web scraping” usually refers to a process that involves automation. When we scrape a website, we write a code that sends a request to the server that’s hosting the page we specified. Our code downloads that page’s source code, just as a browser would. But instead of displaying the page visually, it filters through the page looking for HTML elements we’ve specified, and extracting whatever content we’ve instructed it to extract. For our project, we extract the Date of the notice from the HTML component.

3.4 Pyrogram

Pyrogram is a modern, elegant and easy-to-use Telegram framework written from the ground up in Python and C. It enables you to easily create custom apps for both user and bot identities (bot API alternative) via the MTProto API. In our project, we use Pyrogram to create the interface necessary for sending and receiving messages to the Telegram bot.

3.5 Heroku

Heroku is a cloud platform as a service supporting several programming languages. The Heroku network runs the customer's apps in virtual containers which execute on a reliable runtime environment. Heroku calls these containers "Dynos". In our project, we have integrated heroku as an option to be used for deployment of the code. Also, we consider this as the most efficient and easy to deploy method for the program.

CHAPTER - 4

Environment Variables

An environment variable is a dynamic-named value that can affect the way running processes will behave on a computer. They are part of the environment in which a process runs. They can be used to define the values of certain variables that are integral to the program and might require changes before execution. In our project, there are two types of vars, Optional and Mandatory.

4.1 MANDATORY VARIABLES

4.1.1 ***TG_BOT_TOKEN***

The Telegram Bot's API Token. Obtained from @Botfather .

4.1.2 ***API_ID***

Your Telegram Account's API ID. Obtained from https://my.telegram.org/apps.

4.1.3 ***API_HASH***

Your Telegram Account's API HASH. Obtained from https://my.telegram.org/apps.

4.1.4 ***AUTH_CHANNEL***

Your Telegram Account's unique chat ID Number. Can be obtained from @Rose by sending "/id" in chat.

4.1.5 ***MONGO_URL***

The Connection URL to MongoDB Collection. Can be obtained by creating a cluster on MongoDB and getting the connection url.
NOTE: You MUST enter 0.0.0.0/0 in the allowed ip addresses range in your MongoDB cluster or it might prompt unexpected errors.

4.2 Optional Environment Variables

4.2.1 ***TG_BOT_WORKERS***

Number of Heroku workers to use. *4* is the recommended (and default) amount, but your experience may vary. Note that going crazy with more workers won't necessarily speed up your bot, given the amount of MongoDB data accesses, and the way python asynchronous calls work.

4.2.2 ***COMMM_AND_PRE_FIX***

The command prefix in the bot. Telegram, by default, recommends the use of "/ **"** , which is the default prefix.

4.2.3 ***START_COMMAND***

The command to check if bot is working currently and also for new users to start the bot. Default is "*start*".

4.2.4 ***START_OTHER_USERS_TEXT***

The message that your bot users would see on sending /start command.

4.2.5 ***ONLINE_CHECK_START_TEXT***

The message that the bot administrators can use to check if bot is online.

4.2.6 ***HELP_MEHH***

The message that bot users would see on sending /help command.

4.2.7 ***TZ***

Timezone to make sure time related functions work as required. Default is *Asia/Kolkata.*

4.2.8 ***LOG_FILE_ZZGEVC***

Path to store Log file. Default is *bot/DTUAlertBot.log* .

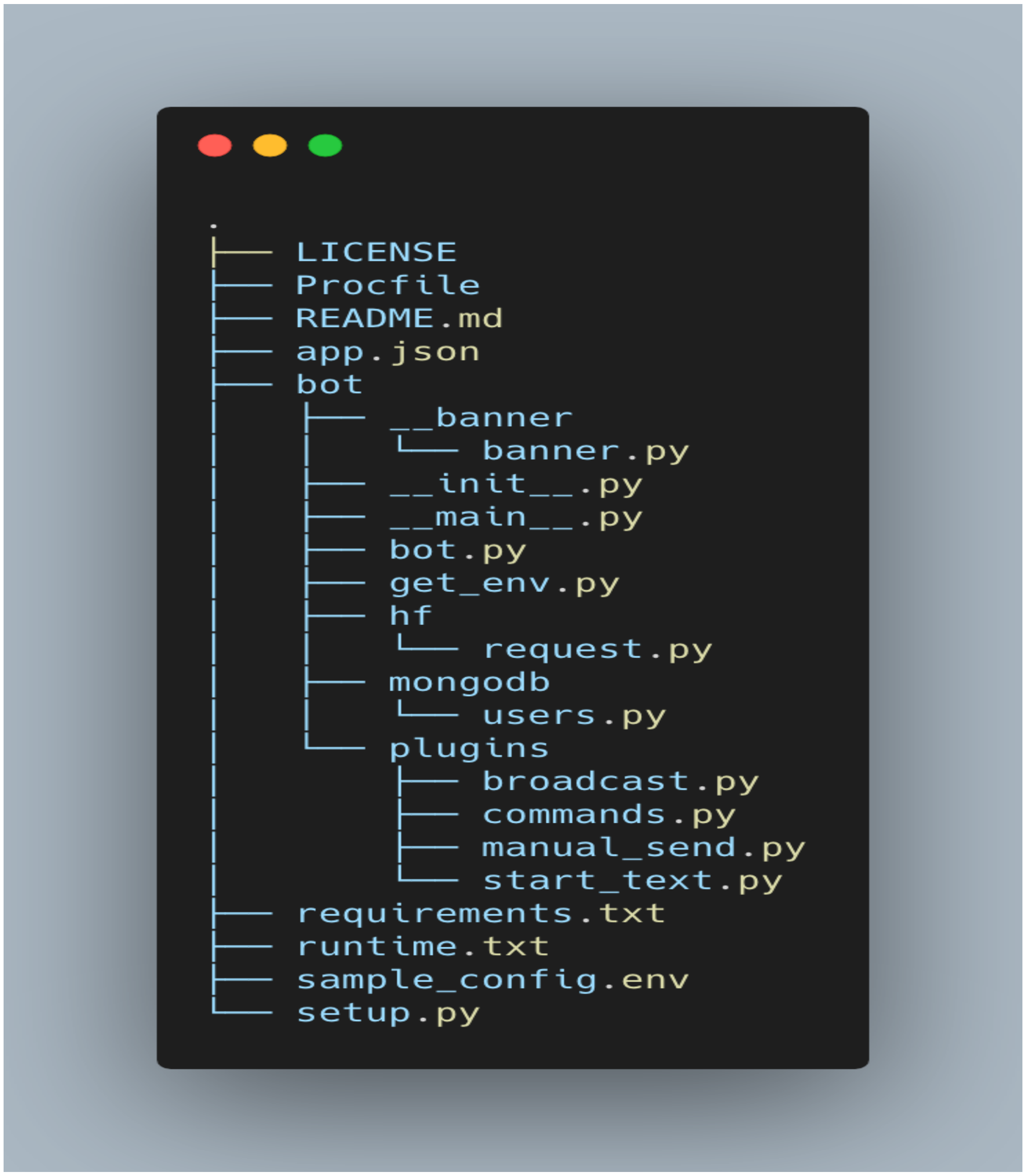
4.2.9 ***REQUEST_INTERVAL***

The time interval between each check on DTU’s official website. Default is 300 seconds.

CHAPTER - 5

Project Tree and Layout

5.1 Project Tree



```
.
├── LICENSE
├── Procfile
├── README.md
├── app.json
├── bot
│   ├── __banner
│   │   └── banner.py
│   ├── __init__.py
│   ├── __main__.py
│   ├── bot.py
│   ├── get_env.py
│   ├── hf
│   │   └── request.py
│   ├── mongodb
│   │   └── users.py
│   └── plugins
│       ├── broadcast.py
│       ├── commands.py
│       ├── manual_send.py
│       └── start_text.py
├── requirements.txt
├── runtime.txt
├── sample_config.env
└── setup.py
```

5.2 External Project Files

5.2.1 **LICENSE**

Public repositories on GitHub are often used to share open source software. For our repository to truly be open source, we'll need to license it so that others are free to use, change, and distribute the software. This project has been licensed under GNU AFFERO GENERAL PUBLIC LICENSE.

5.2.2 **Procfile**

Present for use in Heroku deployment. Heroku apps include a Procfile that specifies the commands that are executed by the app on startup. Each dyno in an app belongs to one of the declared process types, and it executes the startup command associated with that process type.

In our project, the Procfile allows the dyno to execute the command "*python3 -m bot*", which is used to run the bot.

5.2.3 **README.md**

A README file contains information about other files in a directory or archive of computer software. It can be considered an introduction to the project or a documentation of the software.

5.2.4 **app.json**

app.json is a manifest format for describing web apps. It declares environment variables, add-ons, and other information required to run an app on Heroku. This is useful when deploying the app to Heroku.

5.2.5 **requirements.txt**

The ecosystem consisting of your particular installed version of python, plus all the third-party packages ("libraries") it can access (and their precise versions) is called as a Python Environment.
We generate and share requirements.txt files to make it easier for other developers to install the correct versions of the required Python libraries (or "packages") to run the Python code we've written.

5.2.6 **sample_config.env**

This is an example of the config.env file that is required to run the software locally. All the required and optional environment variables have been mentioned in the file. During local deployment, the environment variables can be copied from the *sample_config.env* file to the *config.env* file to configure the environment.

5.2.7 **setup.py**

An automated method to quickly configure and set up the environment variables as well as the requirements and dependencies.

CHAPTER - 6

Core Project Files and Functions

6.1 **bot/__init__.py**

The __init__.py file is required to make Python treat the directories as containing packages; this is done to prevent directories with a common name, such as string, from unintentionally hiding valid modules that occur later on the module search path.

In addition to labeling a directory as a Python package, __init__.py allows us to define any variable at the package level. Doing so is often convenient if a package defines something that will be imported frequently, in an API-like fashion, or the environment variables.

The Environment Variables have been defined and declared in this file. When required, they are imported into the required files respectively. Moreover, the basic configuration for LOGGER, using the logging throughout the project has been defined here.

6.2 **bot/__main__.py**

'__*main*__' is the name of the scope in which top-level code executes. A module’s __name__ is set equal to '__main__' when read from standard input, a script, or from an interactive prompt.

For a package, the same effect can be achieved by including a *__main__.py* module, the contents of which will be executed when the module is run with **-m**.

This is the package which allows our bot to execute and run and create an asynchronous session with Pyrogram.

6.3 **bot/bot.py**

The core of the bot program. The class ‘Bot’ is defined and executed which runs the bot and creates an asynchronous session with Telegram API using the API_ID, API_HASH and TG_BOT_TOKEN variables. Further functions are imported as pluggable modules present in the ‘*bot/plugins*’ directory.

6.4 **bot/get_env.py**

A wrapper to get the required credentials of the environment variables in the declared format. If the called environment variable’s value has not been defined and the *should_prompt* argument is set to true, it will prompt on the CLI to input the variable’s value.

However, this value will be kept in memory for the ongoing execution and will have to be reinstated on the next execution of the program. Hence, it is recommended to define the vars beforehand.

6.5 **bot/__banner/__banner.py**

The package contains the function bannerTop. The function simply contains the ASCII format banner for the program as a string and returns a variable containing the banner as a string.

6.6 **bot/hf/request.py**

The package contains the function request_time which is responsible for web scraping and filtering. The function utilises these python libraries -

- Requests - Requests will allow us to send HTTP/1.1 requests using Python. It can download the HTML code of a website URL and allow us to work with it.

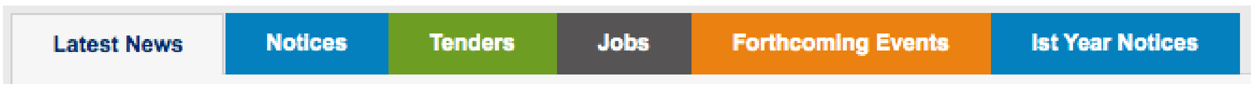
- BeautifulSoup - This library allows us to view and iterate over the HTML code, as well as find the HTML components and elements that we are looking for.

- lxml - This library helps convert the raw HTML code into a proper XML format for better iteration and visualization.

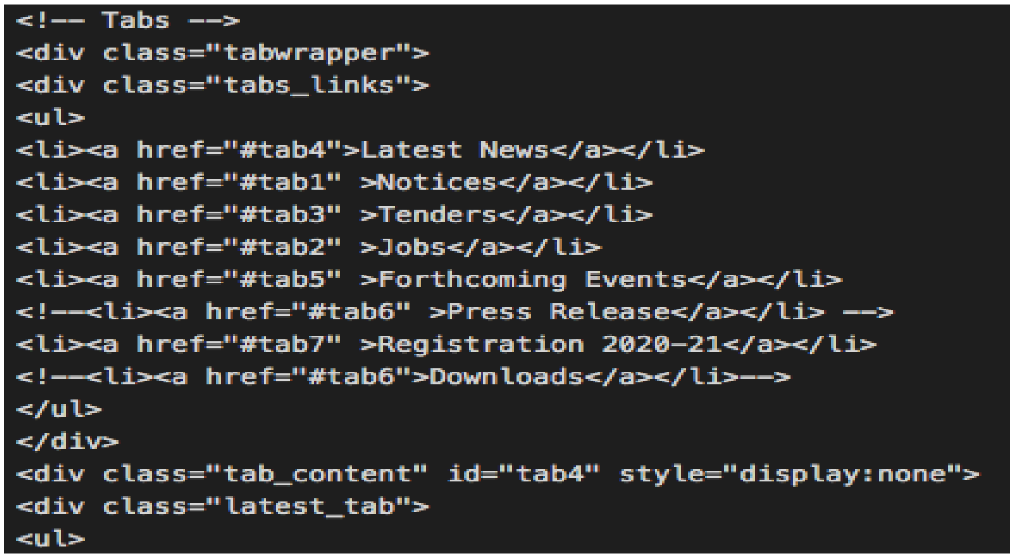
The flowchart for the same can be found at https://whimsical.com/flowchart-request-py-MzWDL9YLMW4FweZLM9HbaY.

The function request_time uses Python's request library to send HTTP GET request to https://dtu.ac.in and download the HTML code for the homepage. Using html component of the lxml library, we can rearrange the HTML code for better iteration and extracting the required content.

In our project, when the bot is executed, the first request made to the website will record the dates of the latest 7 notices under each category and save it in a file with path 'bot/hf/recorded_status.json'. The category here refers to the multiple sections that the notices are uploaded under, namely -



Upon viewing the HTML source code of the page, we can see that each of these categories are labelled as #tab1, #tab2 and such. Hence, we can make use of these classifications to record the top 7 notices under each category in a systematic manner.


```
<!-- Tabs -->
<div class="tabwrapper">
<div class="tabs_links">
<ul>
<li><a href="#tab4">Latest News</a></li>
<li><a href="#tab1" >Notices</a></li>
<li><a href="#tab3" >Tenders</a></li>
<li><a href="#tab2" >Jobs</a></li>
<li><a href="#tab5" >Forthcoming Events</a></li>
<!--<li><a href="#tab6" >Press Release</a></li> -->
<li><a href="#tab7" >Registration 2020-21</a></li>
<!--<li><a href="#tab6">Downloads</a></li>-->
</ul>
</div>
<div class="tab_content" id="tab4" style="display:none">
<div class="latest_tab">
<ul>
```

The dates are stored in a python dictionary and recorded in such a format -
Date.<Tab Number>.<Notice number>: ["*Date*"]
Where,
<Tab Number> denotes the category.
<Notice Number> denote the notice's serial number.
Date denotes the Date of the specific notice.

When calling the request_time next time, the dates of the top 7 notices under each category is recorded in a similar manner. It is then compared to the recorded dates that have been saved in "bot/hf/recorded_status.json" file using the dict_compare function. If there have been any new changes, the status_code is 200. Else, it is 404.

The function request_time returns a Python List variable, return_values after execution of the instructions. If there have not been any new changes, the list is of format -
[<*Check Status Code (**404**)*>, <*Top Notice under category 'Latest News'*>, <*Top Notice's attached link*>, <null>, <null>]

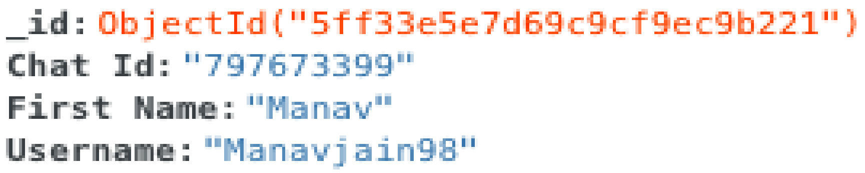
If there have been any new changes, the list is of format -
[<*Check Status Code (**200**)*>, <*Top Notice's Title under category 'Latest News'*>, <*Top Notice's attached link*>, <*New Notice's Title*>, <*New Notice's Link*>, <*New Notice's Category*>]

6.7 **bot/mongodb/users.py**

This package deals with all our operations regarding User Database in MongoDB. MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB.

For our project, The Database is named **DTU** and the collection as **users.** We insert **documents** in **users** collection which is present in the **DTU** Database which is present under our MongoDB Cluster.

The add_client_to_db function is responsible for checking if a user is already present in the database or is a new user. If it is a new user, the details mentioned below are inserted in the **users** collection in a fixed document format.
"Chat Id": "<User's unique Telegram Chat ID>",
"First Name": "<User's First name>",
"Username": "<User's Username, if present >"
An example of a document has been shown below.

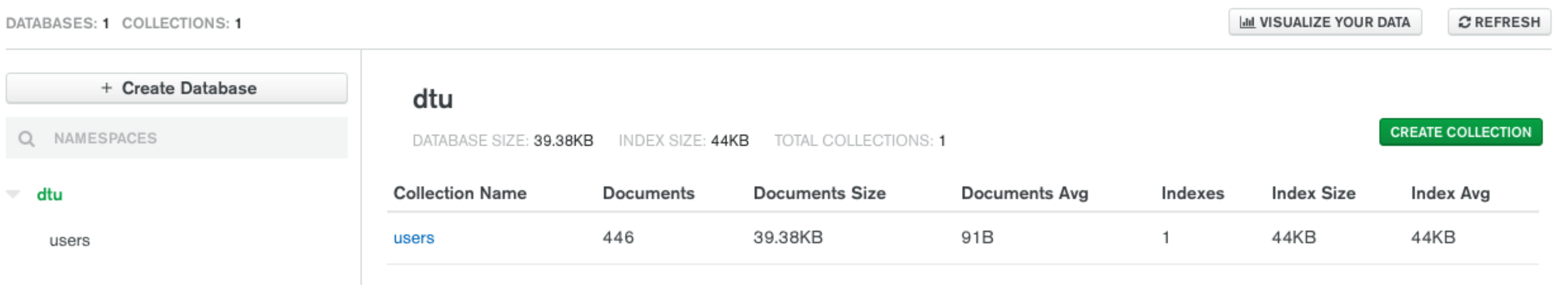


```
_id: ObjectId("5ff33e5e7d69c9cf9ec9b221")
Chat Id: "797673399"
First Name: "Manav"
Username: "Manavjain98"
```

The user_list function is used to get a list of all users' Chat Ids present in the database.

The remove_client_from_db function is used to remove a user using their unique Chat ID.

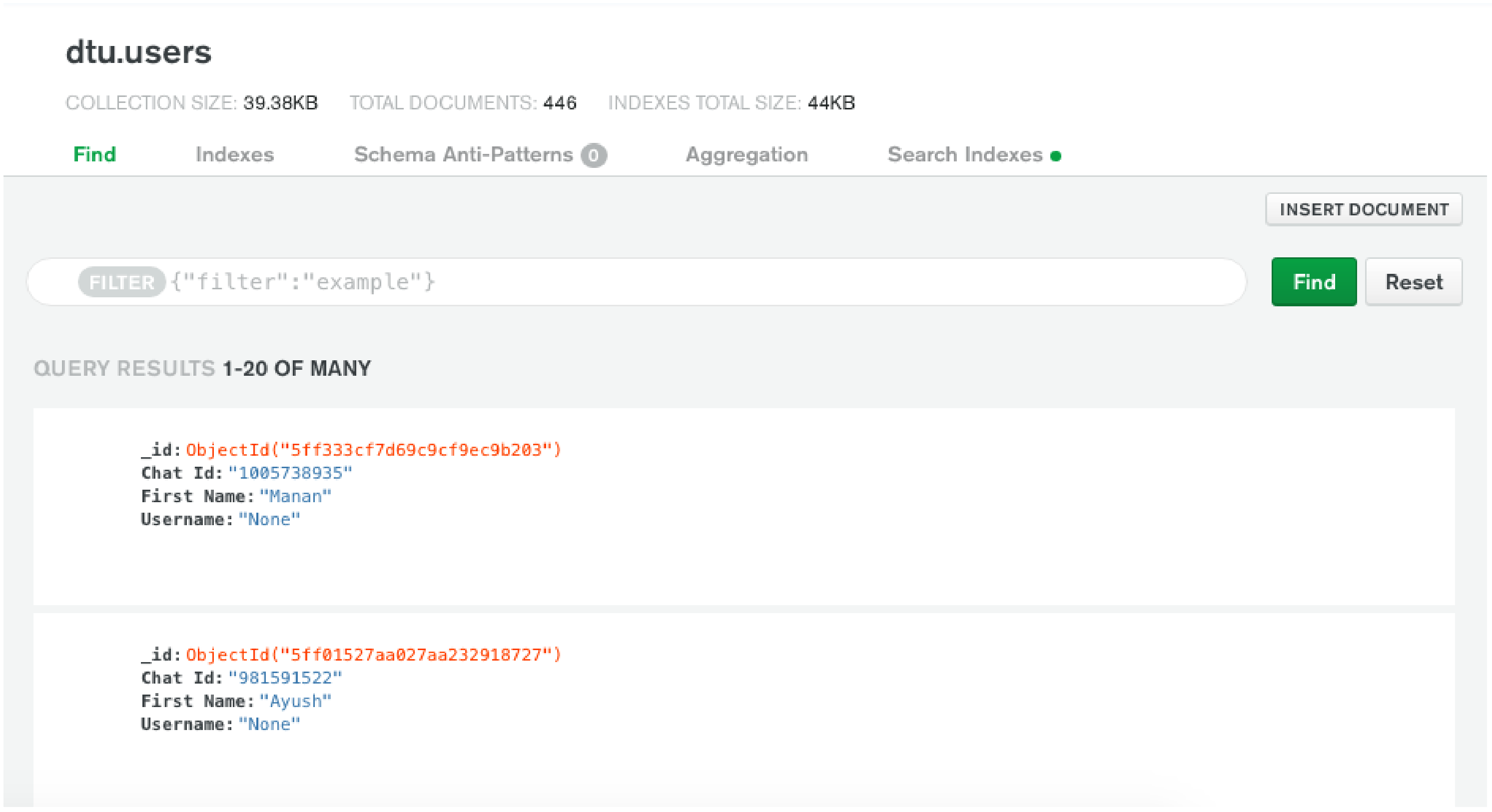
An example of a MongoDB Collection is shown below



dtu

DATABASE SIZE: 39.38KB INDEX SIZE: 44KB TOTAL COLLECTIONS: 1

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
users	446	39.38KB	91B	1	44KB	44KB



dtu.users

COLLECTION SIZE: 39.38KB TOTAL DOCUMENTS: 446 INDEXES TOTAL SIZE: 44KB

{"filter":"example"}

QUERY RESULTS 1-20 OF MANY

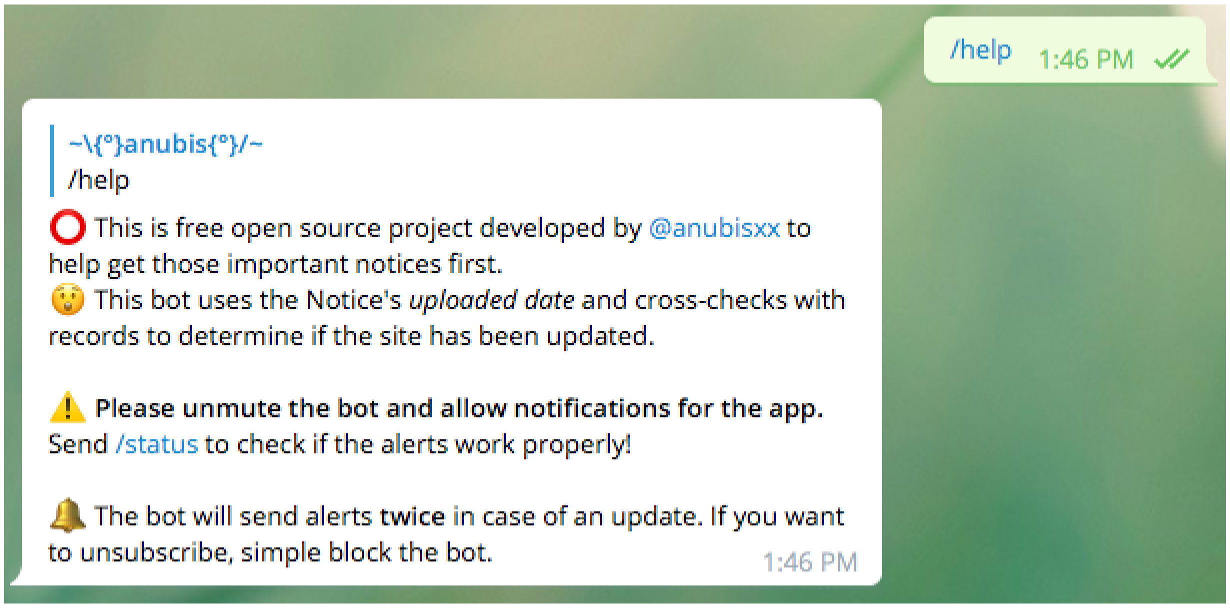
```
_id: ObjectId("5ff333cf7d69c9cf9ec9b203")
Chat Id: "1005738935"
First Name: "Manan"
Username: "None"
```

```
_id: ObjectId("5ff01527aa027aa232918727")
Chat Id: "981591522"
First Name: "Ayush"
Username: "None"
```

6.8 **bot/plugins/commands.py**

This package includes some miscellaneous commands to aid in the functionality of the bot on the front end, that is on Telegram interface.

The function get_this_man_some_help is used to handle the command "**/help**" when sent on bot. This outputs a fixed string present in the Env var **HELP_MEHH**. An example has been provided below.


/help 1:46 PM

~\{°}anubis{°}/~
/help

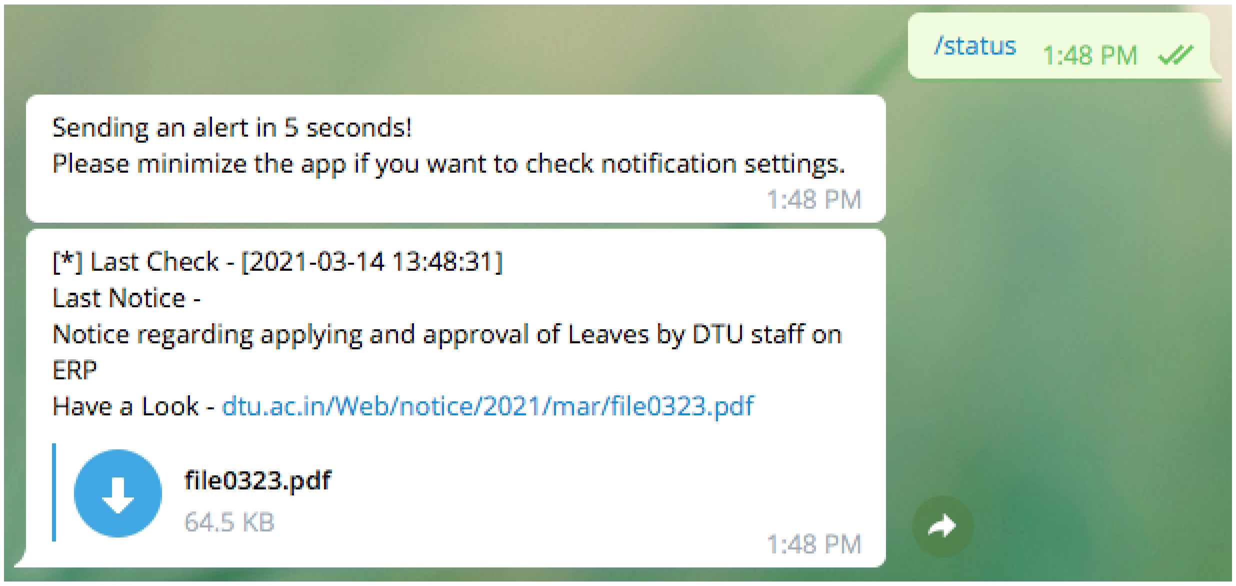
⭕ This is free open source project developed by @anubisxx to help get those important notices first.
😲 This bot uses the Notice's *uploaded date* and cross-checks with records to determine if the site has been updated.

⚠️ **Please unmute the bot and allow notifications for the app.**
Send /status to check if the alerts work properly!

🔔 The bot will send alerts **twice** in case of an update. If you want to unsubscribe, simple block the bot. 1:46 PM

The function tu_ruk_baba_me_dekhta_na is used to handle the command "**/status**" when sent on bot. This command can be used to send a simple message after a few seconds to make sure that the bot is working properly and the notifications can be pushed as intended on the app.

An example has been provided below.



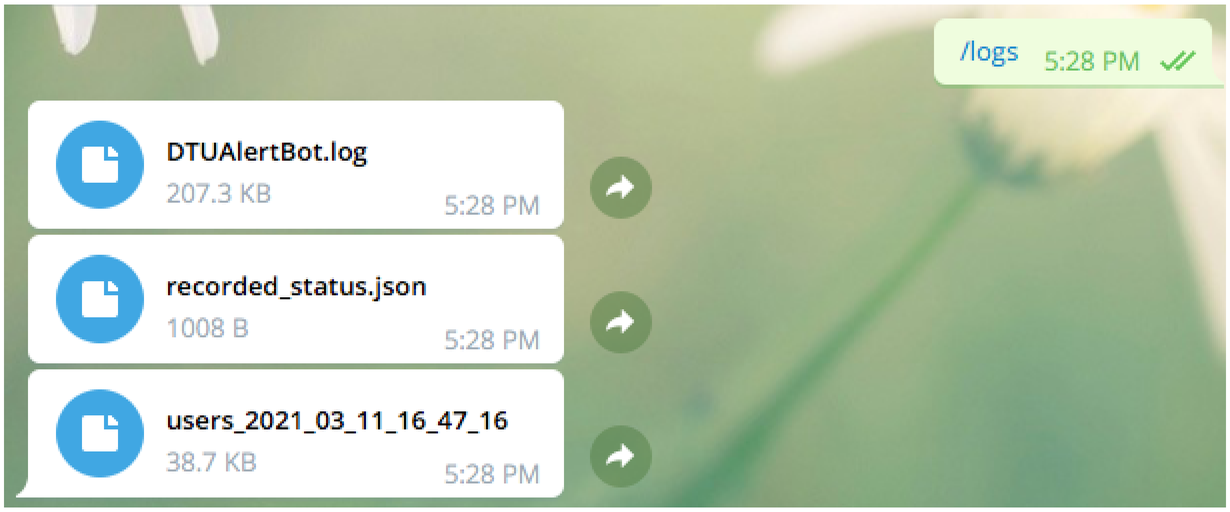
/status 1:48 PM

Sending an alert in 5 seconds!
Please minimize the app if you want to check notification settings.
1:48 PM

[*] Last Check - [2021-03-14 13:48:31]
Last Notice -
Notice regarding applying and approval of Leaves by DTU staff on ERP
Have a Look - dtu.ac.in/Web/notice/2021/mar/file0323.pdf

file0323.pdf
64.5 KB
1:48 PM

The function ye_dekh_kya_hogaya is an admin only command. The command, "/**logs**" can be used by the admin to get the logs file, the current recorded dates and the MongoDB local backup present at path "bot/hf/". This is a kinky feature to monitor the logs when it is inconvenient to access the CLI. An example has been provided below.



/logs 5:28 PM

DTUAlertBot.log
207.3 KB 5:28 PM

recorded_status.json
1008 B 5:28 PM

users_2021_03_11_16_47_16
38.7 KB 5:28 PM

6.9 **bot/plugins/manual_send.py**

This package allows the admin to manually broadcast any important Notice or announcement that might have been skipped by the bot due to downtime, bugs or during maintenance.

The missed_noti function can be used by the bot admin to broadcast a notice to all the users manually through the Telegram Bot Chat interface. Simply sending a command "**/send**" along with the required arguments will allow the bot to send the notification to all users.

The command has to be in this format -
/send | <Notice's attached URL> | <Notice's Title> | <Notice's Category Name>

The command will follow the usual broadcast method except that all the required values obtained from the request_time function from "bot/hf/request.py" have been provided manually instead.

6.10 **bot/plugins/broadcast.py**

This package is responsible for broadcasting and sending notifications on Telegram whenever the website has been updated.

The Flowchart for the instructions can be found at https://whimsical.com/flowchart-broadcast-py-LYPjcpVLRpH84cFyDApvwe

This also includes the main loop of our program, to call the request_time() function at every fixed interval. Here, we make use of Python's Threading library that can help create threads at a mentioned interval that has been configured in the vars as **REQUEST_INTERVAL.**

We make use of sendtelegram to send messages to a user using their Chat ID that has been recorded in the Database. If the user has blocked our bot, It will return an error code, 403 indicating that the user has blocked the bot. In this case, we will append this user's Chat ID to our variable list, failed_users. Once the broadcast has been sent to all recipients, we will remove all the users present in failed_users from our database.

Since we're dealing with our Database management, if there happen to be any bugs, we might end up losing the full database! Thus, we've used Python library **os** to run the *mongoexport* command, which will backup our database **before** starting the broadcast. Once the broadcast has been sent, the local backup file will be logged and sent to the admin's Chat in Telegram.

CHAPTER - 7

Source Code, Deployment and Testing

7.1 Source Code

This project has been made open source and uploaded to Github and can be found at https://github.com/Dark-Princ3/DTU-Alert-Bot. The Project has been licensed under GNU Affero General Public License v3.0, the details of which can be found here.

7.2 Demo / Deployed Bot

With this project having been developed a while ago, We have already deployed and created a bot for Practical usage. The demo bot can be found on Telegram with username @DTUAlertBot . With over 450 people already subscribed for notifications, we can say that this project indeed has technical and practical feasibility.

7.3 Modes of Deployment

In this section, we will have a look at the various methods available to deploy and host the project code on different platforms. We will consider that all the Mandatory Environment Variables mentioned above in Chapter-4 have been prepared.

7.3.1 On a Heroku Server

Although there are various methods available to deploy the code to Heroku, the easiest has been mentioned here. Firstly, we need a Heroku Account.
Next, we can use this hyperlink to get to the deployment page. Next, configure the Environment Variables as prompted and click on the "Deploy App" button at the bottom. That's it!
It's that simple to configure and deploy the bot to a Heroku Server, even a layman can try it out.

7.3.2 On a GNU / Windows / Linux System

We assume that the Requirements mentioned in Chapter 2 have been installed and are ready.

1. Download the source code zip from here.
2. Extract the downloaded source code.
3. Open a Command Line Interface (CLI) session.
4. Enter *cd <Code Directory>*.
5. Make sure you are in root directory of the code.
6. Enter *python3 setup.py*
7. Enter the values for Environment Variables as prompted.
8. Enter *python3 -m bot* to execute the bot.

7.3.3 On an Android Device

1. Download Termux application from Play Store.
2. Open Termux and enter *termux-setup-storage*. Allow Storage permissions.
3. Enter *pkg install python git libxml2 libxslt clang*.
4. Enter *python3 -m venv venv*.
5. Enter *. ./venv/bin/activate*.
6. Enter *cd /sdcard*.
7. Enter *git clone https://github.com/Dark-Princ3/DTU-Alert-Bot*.
8. Enter *cd DTU-Alert-Bot*.
9. Enter *python3 setup.py*.
10. Enter the 'mandatory' environment variables as prompted.
11. Once setup is done, enter *python3 -m bot*.

7.4 Values for Mandatory Environment Variables for Testing

For the purpose of quickly testing out the project and bot without going through the hassle to collect values for vars, we've created a dummy environment and resources for quick testing purposes.
These values shall remain working for one month from the publishing of this report.

- **TG_BOT_TOKEN** - 1543512697:AAFXlVT294T2oLQC35TuNqtE3XWQ7wx8P80

- **API_ID -**
  1255294

- **API_HASH -**
  074212fca27cd02475ba8b82a92fa8ff

- **AUTH_CHANNEL -**
  Your own Telegram ID. Check Chapter 2 for more information.

- **MONGO_URL -**
  mongodb+srv://dbUser:dbPass@bottesting.un7lk.mongodb.net/BotTesting?retryWrites=true&w=majority

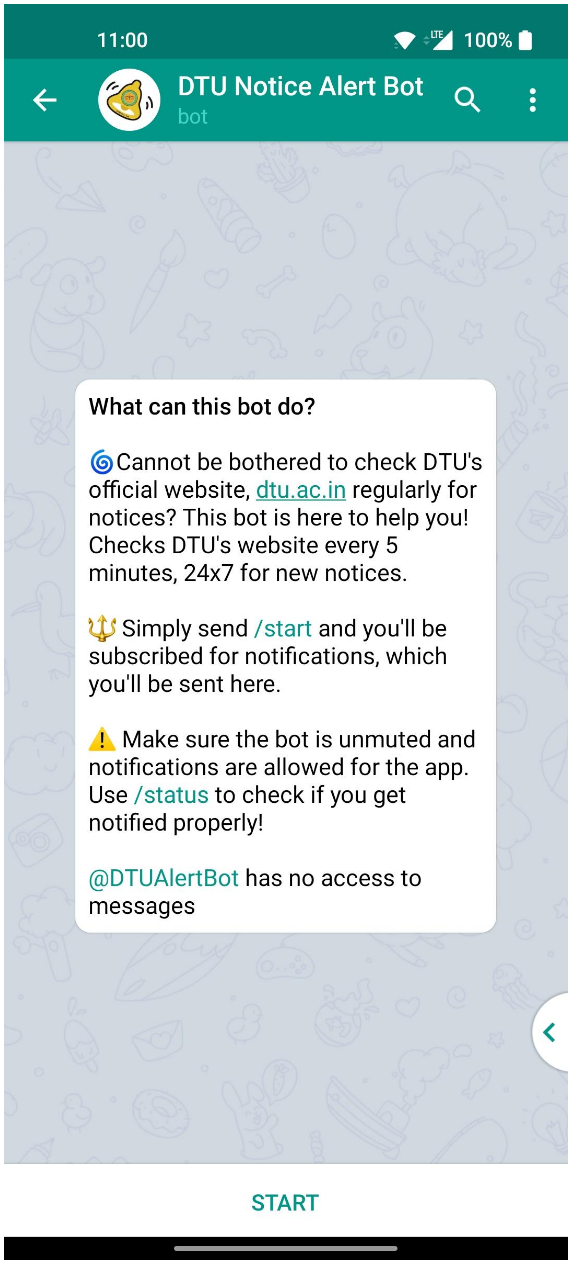
The Telegram Bot with the above mentioned token can be found at @DTUTestingBot.

CHAPTER - 8

Snapshots

Front End

<u>Demo Bot Profile</u>


11:00 100%

DTU Notice Alert Bot
bot

What can this bot do?

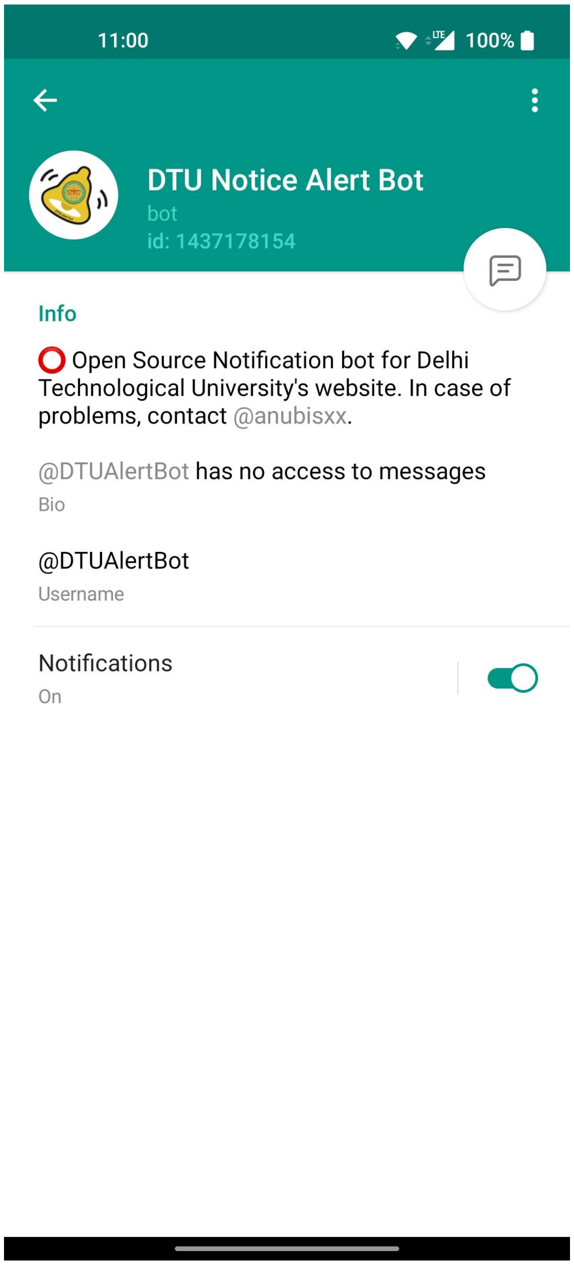
🌀Cannot be bothered to check DTU's official website, dtu.ac.in regularly for notices? This bot is here to help you! Checks DTU's website every 5 minutes, 24x7 for new notices.

🔱 Simply send /start and you'll be subscribed for notifications, which you'll be sent here.

⚠️ Make sure the bot is unmuted and notifications are allowed for the app. Use /status to check if you get notified properly!

@DTUAlertBot has no access to messages

START


11:00 100%

DTU Notice Alert Bot
bot
id: 1437178154

Info

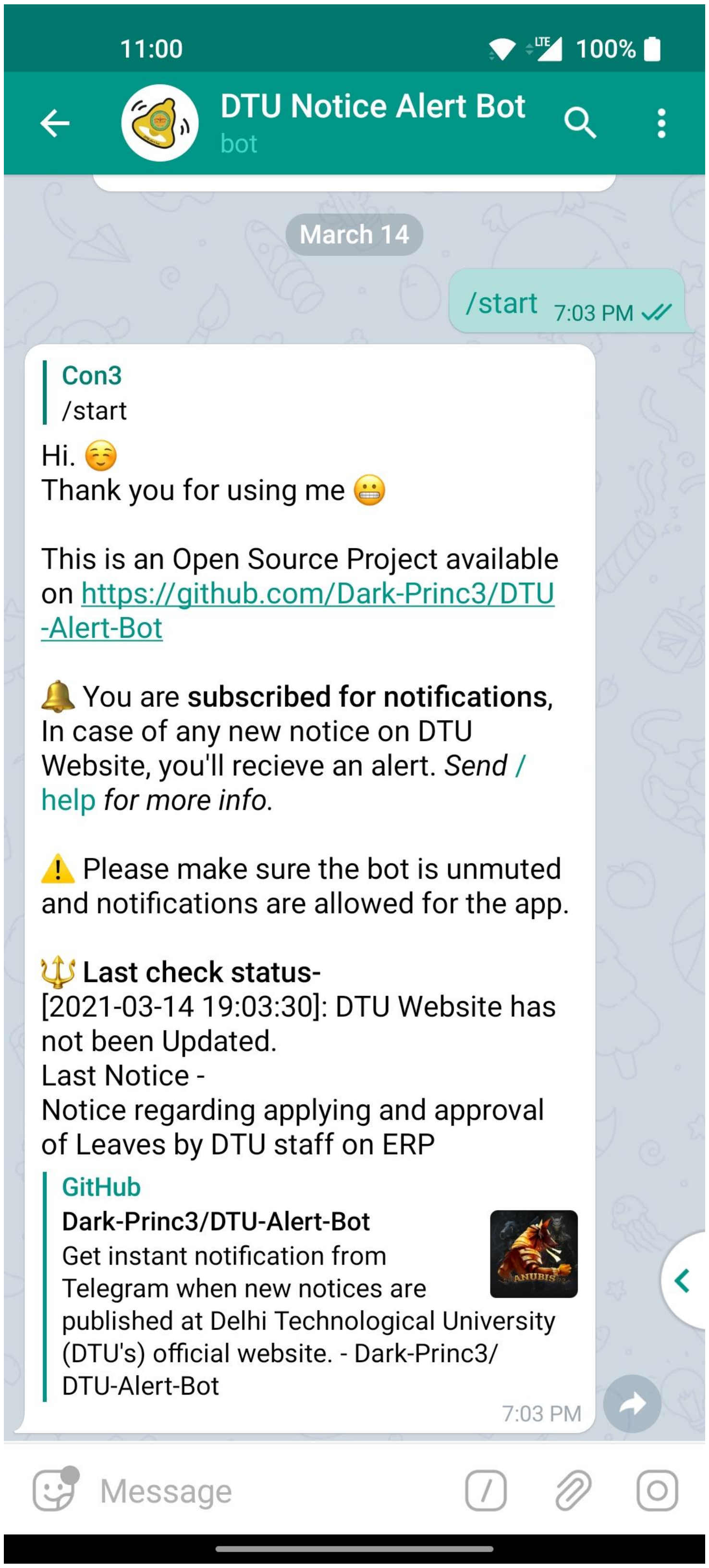
⭕ Open Source Notification bot for Delhi Technological University's website. In case of problems, contact @anubisxx.

@DTUAlertBot has no access to messages
Bio

@DTUAlertBot
Username

Notifications
On

/start command by user


11:00 100%

DTU Notice Alert Bot
bot

March 14

/start 7:03 PM

Con3
/start

Hi. ☺️
Thank you for using me 😬

This is an Open Source Project available on https://github.com/Dark-Princ3/DTU-Alert-Bot

🔔 You are **subscribed for notifications**, In case of any new notice on DTU Website, you'll recieve an alert. *Send* /help *for more info.*

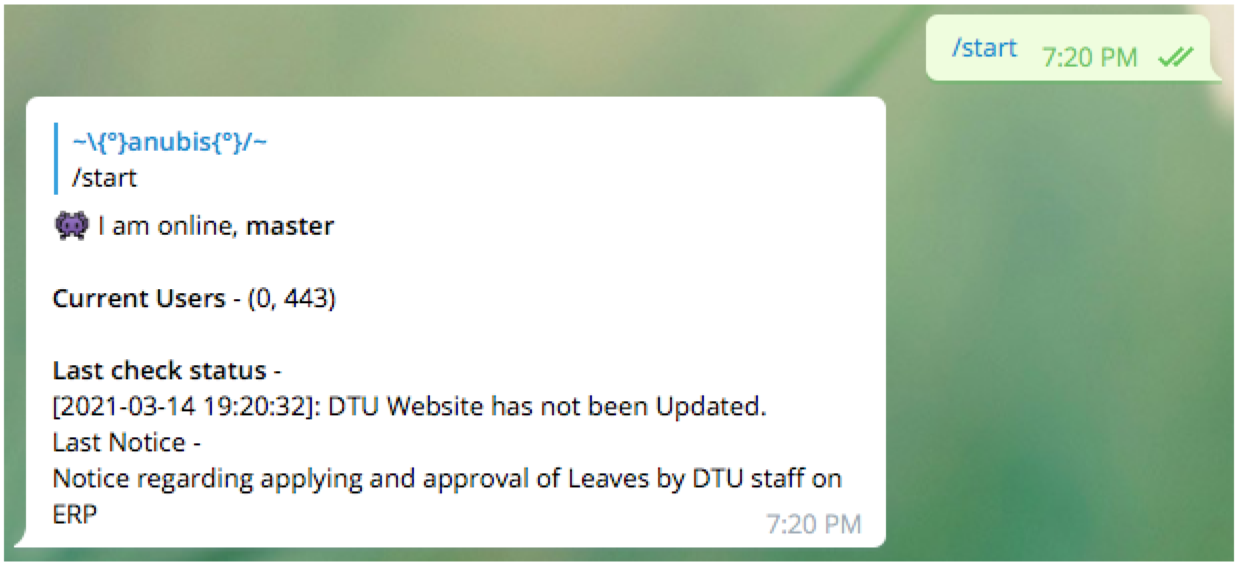
⚠️ Please make sure the bot is unmuted and notifications are allowed for the app.

🔱 **Last check status-**
[2021-03-14 19:03:30]: DTU Website has not been Updated.
Last Notice -
Notice regarding applying and approval of Leaves by DTU staff on ERP

GitHub
Dark-Princ3/DTU-Alert-Bot
Get instant notification from Telegram when new notices are published at Delhi Technological University (DTU's) official website. - Dark-Princ3/DTU-Alert-Bot
7:03 PM

Message

/start command by admin



/start 7:20 PM

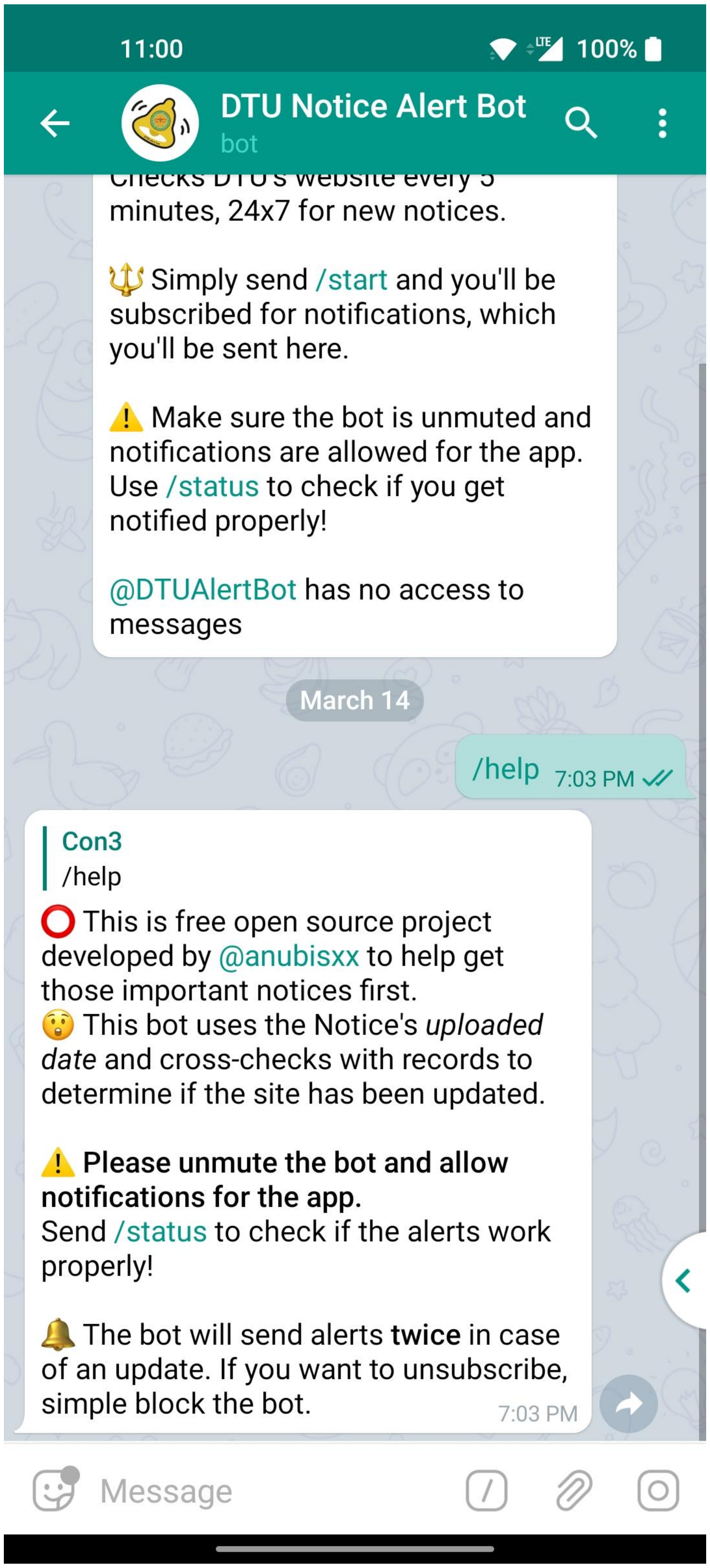
~\{°}anubis{°}/~
/start

👾 I am online, **master**

Current Users - (0, 443)

Last check status -
[2021-03-14 19:20:32]: DTU Website has not been Updated.
Last Notice -
Notice regarding applying and approval of Leaves by DTU staff on ERP
7:20 PM

/help command



Checks DTU's website every 5 minutes, 24x7 for new notices.

🔱 Simply send /start and you'll be subscribed for notifications, which you'll be sent here.

⚠️ Make sure the bot is unmuted and notifications are allowed for the app. Use /status to check if you get notified properly!

@DTUAlertBot has no access to messages

March 14

/help 7:03 PM

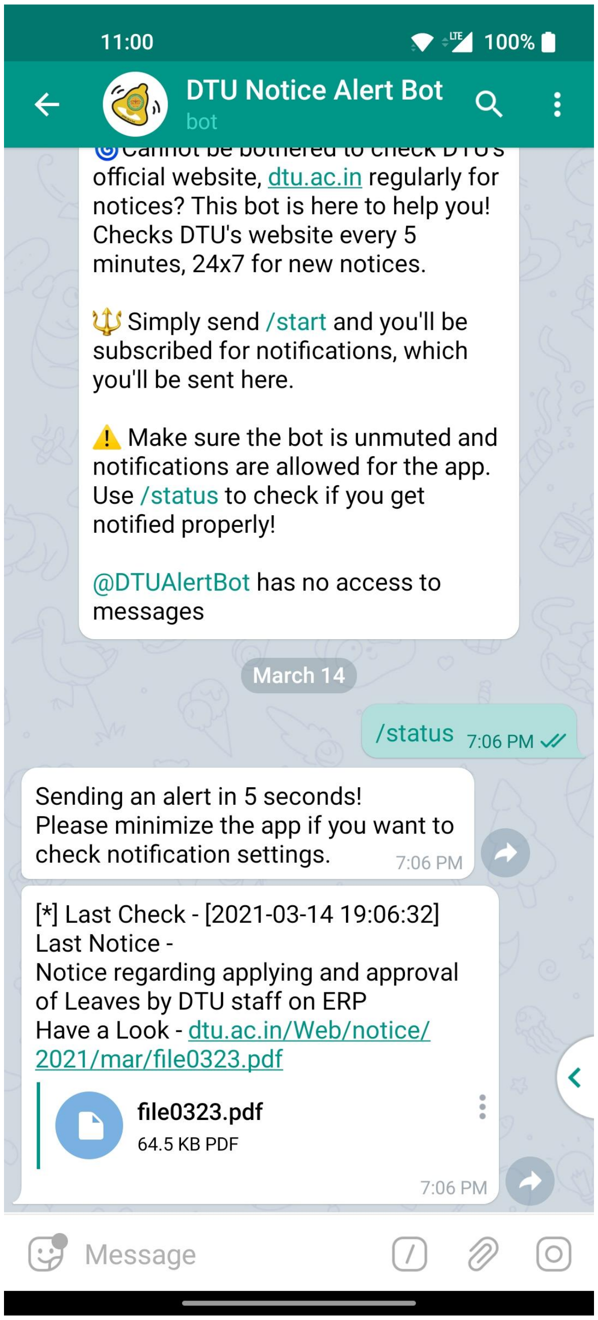
Con3
/help

⭕ This is free open source project developed by @anubisxx to help get those important notices first.
😲 This bot uses the Notice's *uploaded date* and cross-checks with records to determine if the site has been updated.

⚠️ **Please unmute the bot and allow notifications for the app.**
Send /status to check if the alerts work properly!

🔔 The bot will send alerts **twice** in case of an update. If you want to unsubscribe, simple block the bot. 7:03 PM

/status command


11:00 100%

DTU Notice Alert Bot
bot

Cannot be bothered to check DTU's official website, dtu.ac.in regularly for notices? This bot is here to help you! Checks DTU's website every 5 minutes, 24x7 for new notices.

🔱 Simply send /start and you'll be subscribed for notifications, which you'll be sent here.

⚠️ Make sure the bot is unmuted and notifications are allowed for the app. Use /status to check if you get notified properly!

@DTUAlertBot has no access to messages

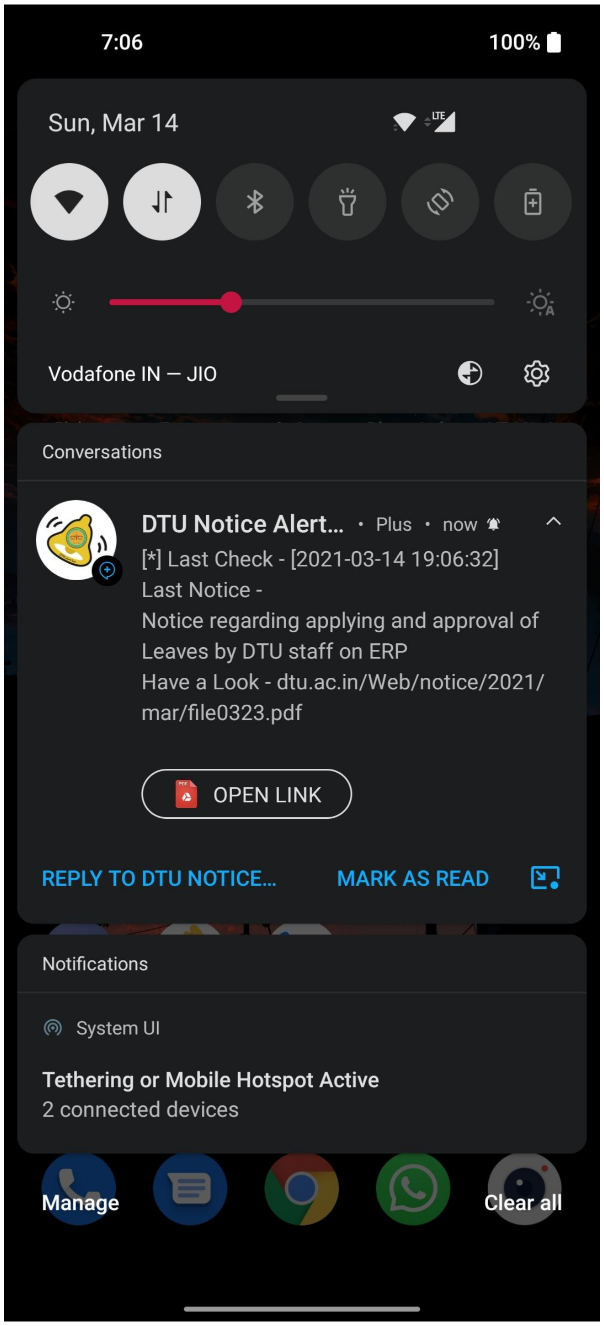
March 14

/status 7:06 PM

Sending an alert in 5 seconds!
Please minimize the app if you want to check notification settings. 7:06 PM

[*] Last Check - [2021-03-14 19:06:32]
Last Notice -
Notice regarding applying and approval of Leaves by DTU staff on ERP
Have a Look - dtu.ac.in/Web/notice/2021/mar/file0323.pdf

file0323.pdf
64.5 KB PDF
7:06 PM


7:06 100%

Sun, Mar 14

Vodafone IN — JIO

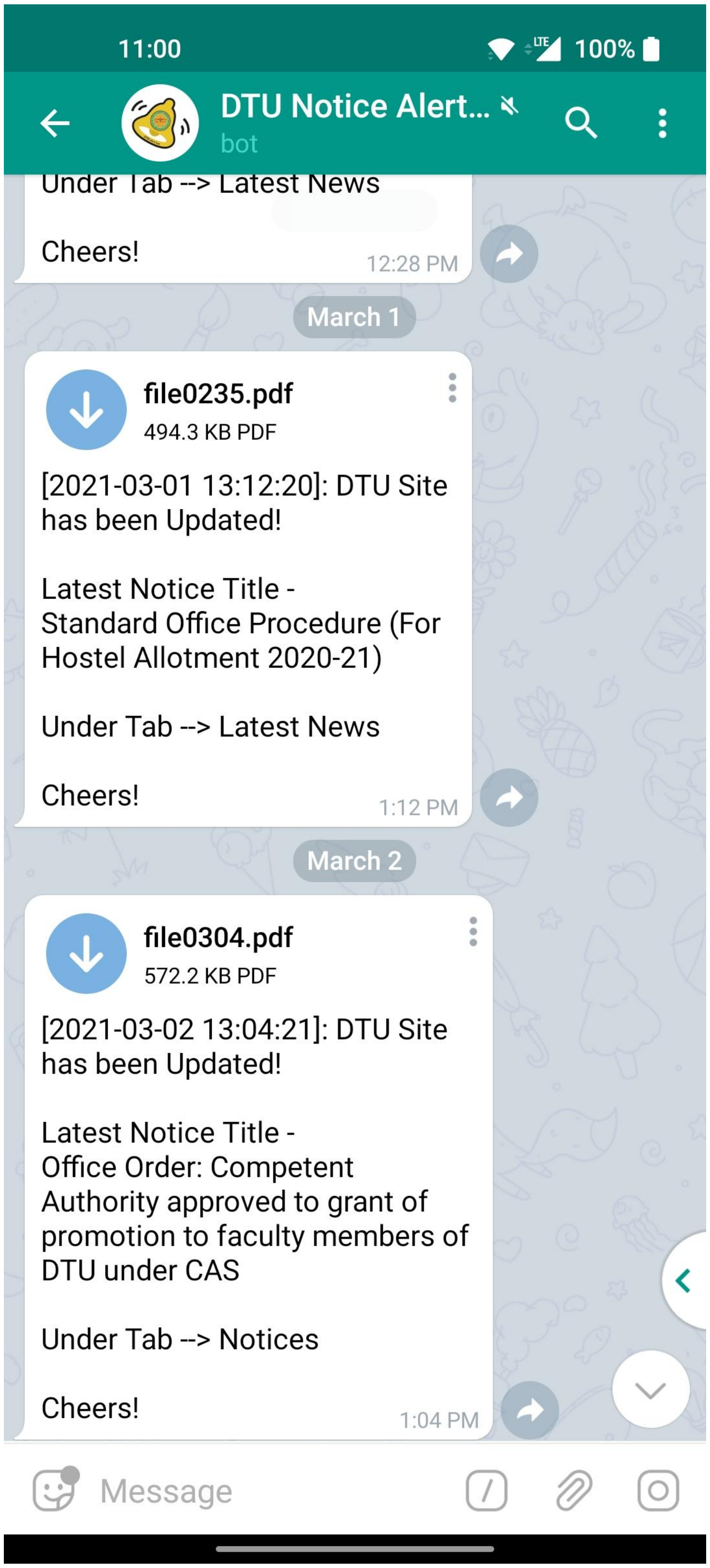
Conversations

DTU Notice Alert... • Plus • now
[*] Last Check - [2021-03-14 19:06:32]
Last Notice -
Notice regarding applying and approval of Leaves by DTU staff on ERP
Have a Look - dtu.ac.in/Web/notice/2021/mar/file0323.pdf

Notifications

System UI
Tethering or Mobile Hotspot Active
2 connected devices

Sample Notice Alert



11:00 100%

DTU Notice Alert...
bot

Under Tab --> Latest News

Cheers! 12:28 PM

March 1

file0235.pdf
494.3 KB PDF

[2021-03-01 13:12:20]: DTU Site has been Updated!

Latest Notice Title -
Standard Office Procedure (For Hostel Allotment 2020-21)

Under Tab --> Latest News

Cheers! 1:12 PM

March 2

file0304.pdf
572.2 KB PDF

[2021-03-02 13:04:21]: DTU Site has been Updated!

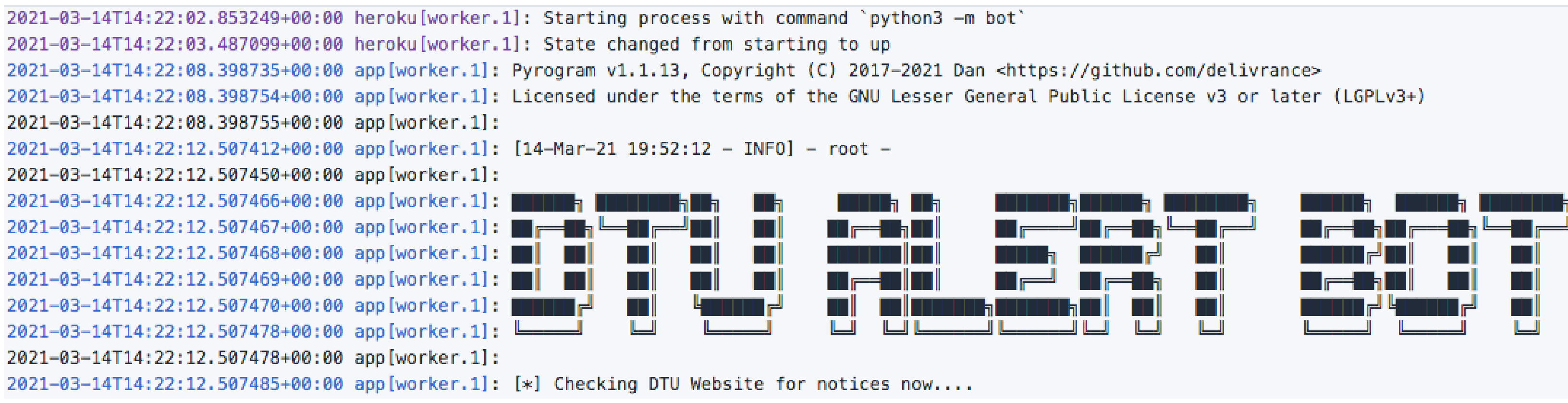
Latest Notice Title -
Office Order: Competent Authority approved to grant of promotion to faculty members of DTU under CAS

Under Tab --> Notices

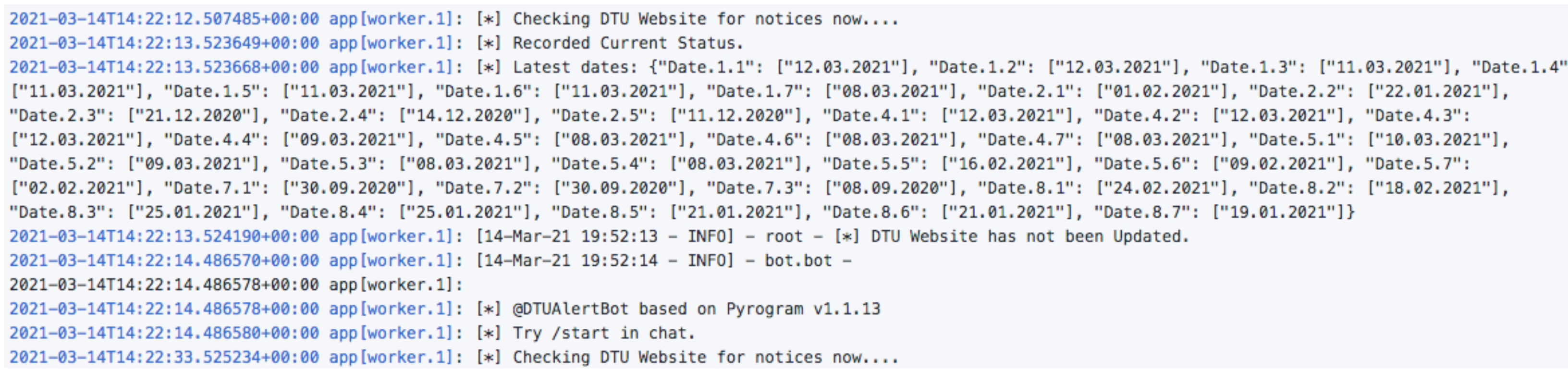
Cheers! 1:04 PM

Back End

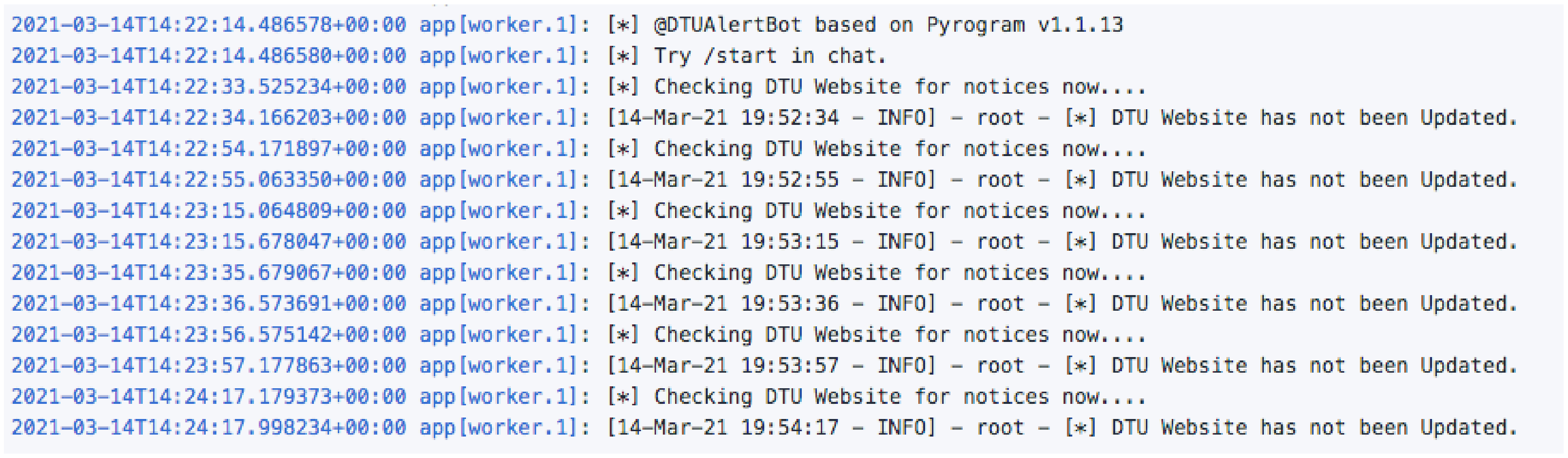
Heroku Logs



```
2021-03-14T14:22:02.853249+00:00 heroku[worker.1]: Starting process with command `python3 -m bot`
2021-03-14T14:22:03.487099+00:00 heroku[worker.1]: State changed from starting to up
2021-03-14T14:22:08.398735+00:00 app[worker.1]: Pyrogram v1.1.13, Copyright (C) 2017-2021 Dan <https://github.com/delivrance>
2021-03-14T14:22:08.398754+00:00 app[worker.1]: Licensed under the terms of the GNU Lesser General Public License v3 or later (LGPLv3+)
2021-03-14T14:22:08.398755+00:00 app[worker.1]:
2021-03-14T14:22:12.507412+00:00 app[worker.1]: [14-Mar-21 19:52:12 - INFO] - root -
2021-03-14T14:22:12.507450+00:00 app[worker.1]:
2021-03-14T14:22:12.507466+00:00 app[worker.1]: ██████╗ ████████╗██╗   ██╗     █████╗ ██╗     ███████╗██████╗ ████████╗    ██████╗  ██████╗ ████████╗
2021-03-14T14:22:12.507467+00:00 app[worker.1]: ██╔══██╗╚══██╔══╝██║   ██║    ██╔══██╗██║     ██╔════╝██╔══██╗╚══██╔══╝    ██╔══██╗██╔═══██╗╚══██╔══╝
2021-03-14T14:22:12.507468+00:00 app[worker.1]: ██║  ██║   ██║   ██║   ██║    ███████║██║     █████╗  ██████╔╝   ██║       ██████╔╝██║   ██║   ██║
2021-03-14T14:22:12.507469+00:00 app[worker.1]: ██║  ██║   ██║   ██║   ██║    ██╔══██║██║     ██╔══╝  ██╔══██╗   ██║       ██╔══██╗██║   ██║   ██║
2021-03-14T14:22:12.507470+00:00 app[worker.1]: ██████╔╝   ██║   ╚██████╔╝    ██║  ██║███████╗███████╗██║  ██║   ██║       ██████╔╝╚██████╔╝   ██║
2021-03-14T14:22:12.507478+00:00 app[worker.1]: ╚═════╝    ╚═╝    ╚═════╝     ╚═╝  ╚═╝╚══════╝╚══════╝╚═╝  ╚═╝   ╚═╝       ╚═════╝  ╚═════╝    ╚═╝
2021-03-14T14:22:12.507478+00:00 app[worker.1]:
2021-03-14T14:22:12.507485+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
```

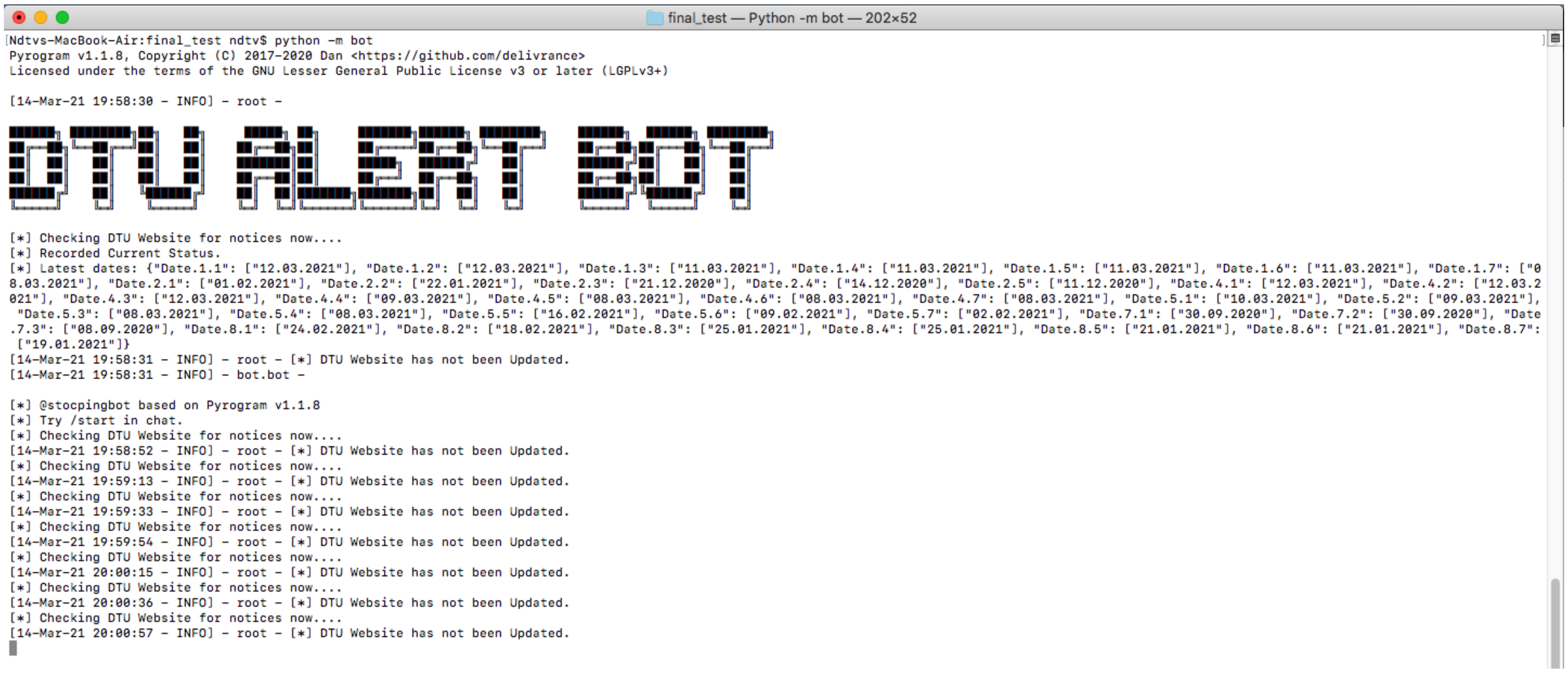


```
2021-03-14T14:22:12.507485+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:22:13.523649+00:00 app[worker.1]: [*] Recorded Current Status.
2021-03-14T14:22:13.523668+00:00 app[worker.1]: [*] Latest dates: {"Date.1.1": ["12.03.2021"], "Date.1.2": ["12.03.2021"], "Date.1.3": ["11.03.2021"], "Date.1.4"
["11.03.2021"], "Date.1.5": ["11.03.2021"], "Date.1.6": ["11.03.2021"], "Date.1.7": ["08.03.2021"], "Date.2.1": ["01.02.2021"], "Date.2.2": ["22.01.2021"],
"Date.2.3": ["21.12.2020"], "Date.2.4": ["14.12.2020"], "Date.2.5": ["11.12.2020"], "Date.4.1": ["12.03.2021"], "Date.4.2": ["12.03.2021"], "Date.4.3":
["12.03.2021"], "Date.4.4": ["09.03.2021"], "Date.4.5": ["08.03.2021"], "Date.4.6": ["08.03.2021"], "Date.4.7": ["08.03.2021"], "Date.5.1": ["10.03.2021"],
"Date.5.2": ["09.03.2021"], "Date.5.3": ["08.03.2021"], "Date.5.4": ["08.03.2021"], "Date.5.5": ["16.02.2021"], "Date.5.6": ["09.02.2021"], "Date.5.7":
["02.02.2021"], "Date.7.1": ["30.09.2020"], "Date.7.2": ["30.09.2020"], "Date.7.3": ["08.09.2020"], "Date.8.1": ["24.02.2021"], "Date.8.2": ["18.02.2021"],
"Date.8.3": ["25.01.2021"], "Date.8.4": ["25.01.2021"], "Date.8.5": ["21.01.2021"], "Date.8.6": ["21.01.2021"], "Date.8.7": ["19.01.2021"]}
2021-03-14T14:22:13.524190+00:00 app[worker.1]: [14-Mar-21 19:52:13 - INFO] - root - [*] DTU Website has not been Updated.
2021-03-14T14:22:14.486570+00:00 app[worker.1]: [14-Mar-21 19:52:14 - INFO] - bot.bot -
2021-03-14T14:22:14.486578+00:00 app[worker.1]:
2021-03-14T14:22:14.486578+00:00 app[worker.1]: [*] @DTUAlertBot based on Pyrogram v1.1.13
2021-03-14T14:22:14.486580+00:00 app[worker.1]: [*] Try /start in chat.
2021-03-14T14:22:33.525234+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
```



```
2021-03-14T14:22:14.486578+00:00 app[worker.1]: [*] @DTUAlertBot based on Pyrogram v1.1.13
2021-03-14T14:22:14.486580+00:00 app[worker.1]: [*] Try /start in chat.
2021-03-14T14:22:33.525234+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:22:34.166203+00:00 app[worker.1]: [14-Mar-21 19:52:34 - INFO] - root - [*] DTU Website has not been Updated.
2021-03-14T14:22:54.171897+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:22:55.063350+00:00 app[worker.1]: [14-Mar-21 19:52:55 - INFO] - root - [*] DTU Website has not been Updated.
2021-03-14T14:23:15.064809+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:23:15.678047+00:00 app[worker.1]: [14-Mar-21 19:53:15 - INFO] - root - [*] DTU Website has not been Updated.
2021-03-14T14:23:35.679067+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:23:36.573691+00:00 app[worker.1]: [14-Mar-21 19:53:36 - INFO] - root - [*] DTU Website has not been Updated.
2021-03-14T14:23:56.575142+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:23:57.177863+00:00 app[worker.1]: [14-Mar-21 19:53:57 - INFO] - root - [*] DTU Website has not been Updated.
2021-03-14T14:24:17.179373+00:00 app[worker.1]: [*] Checking DTU Website for notices now....
2021-03-14T14:24:17.998234+00:00 app[worker.1]: [14-Mar-21 19:54:17 - INFO] - root - [*] DTU Website has not been Updated.
```

Hosting on Macintosh Terminal



```
Ndtvs-MacBook-Air:final_test ndtv$ python -m bot
Pyrogram v1.1.8, Copyright (C) 2017-2020 Dan <https://github.com/delivrance>
Licensed under the terms of the GNU Lesser General Public License v3 or later (LGPLv3+)

[14-Mar-21 19:58:30 - INFO] - root -

DTU ALERT BOT

[*] Checking DTU Website for notices now....
[*] Recorded Current Status.
[*] Latest dates: {"Date.1.1": ["12.03.2021"], "Date.1.2": ["12.03.2021"], "Date.1.3": ["11.03.2021"], "Date.1.4": ["11.03.2021"], "Date.1.5": ["11.03.2021"], "Date.1.6": ["11.03.2021"], "Date.1.7": ["0
8.03.2021"], "Date.2.1": ["01.02.2021"], "Date.2.2": ["22.01.2021"], "Date.2.3": ["21.12.2020"], "Date.2.4": ["14.12.2020"], "Date.2.5": ["11.12.2020"], "Date.4.1": ["12.03.2021"], "Date.4.2": ["12.03.2
021"], "Date.4.3": ["12.03.2021"], "Date.4.4": ["09.03.2021"], "Date.4.5": ["08.03.2021"], "Date.4.6": ["08.03.2021"], "Date.4.7": ["08.03.2021"], "Date.5.1": ["10.03.2021"], "Date.5.2": ["09.03.2021"],
 "Date.5.3": ["08.03.2021"], "Date.5.4": ["08.03.2021"], "Date.5.5": ["16.02.2021"], "Date.5.6": ["09.02.2021"], "Date.5.7": ["02.02.2021"], "Date.7.1": ["30.09.2020"], "Date.7.2": ["30.09.2020"], "Date
.7.3": ["08.09.2020"], "Date.8.1": ["24.02.2021"], "Date.8.2": ["18.02.2021"], "Date.8.3": ["25.01.2021"], "Date.8.4": ["25.01.2021"], "Date.8.5": ["21.01.2021"], "Date.8.6": ["21.01.2021"], "Date.8.7":
 ["19.01.2021"]}
[14-Mar-21 19:58:31 - INFO] - root - [*] DTU Website has not been Updated.
[14-Mar-21 19:58:31 - INFO] - bot.bot -

[*] @stocpingbot based on Pyrogram v1.1.8
[*] Try /start in chat.
[*] Checking DTU Website for notices now....
[14-Mar-21 19:58:52 - INFO] - root - [*] DTU Website has not been Updated.
[*] Checking DTU Website for notices now....
[14-Mar-21 19:59:13 - INFO] - root - [*] DTU Website has not been Updated.
[*] Checking DTU Website for notices now....
[14-Mar-21 19:59:33 - INFO] - root - [*] DTU Website has not been Updated.
[*] Checking DTU Website for notices now....
[14-Mar-21 19:59:54 - INFO] - root - [*] DTU Website has not been Updated.
[*] Checking DTU Website for notices now....
[14-Mar-21 20:00:15 - INFO] - root - [*] DTU Website has not been Updated.
[*] Checking DTU Website for notices now....
[14-Mar-21 20:00:36 - INFO] - root - [*] DTU Website has not been Updated.
[*] Checking DTU Website for notices now....
[14-Mar-21 20:00:57 - INFO] - root - [*] DTU Website has not been Updated.
```

CHAPTER - 9

References

1. https://stackoverflow.com/
2. https://github.com/Dark-Princ3/X-tra-Telegram
3. https://www.geeksforgeeks.org/
4. https://docs.python.org/
5. https://core.telegram.org/#bot-api
6. https://devcenter.heroku.com/articles/procfile
7. https://www.w3schools.com/
8. https://www.tutorialspoint.com/
9. https://www.mongodb.org/
10. http://docs.pyrogram.org/
11. https://heroku.com/
12. https://requests.readthedocs.io/
13. https://lxml.de/