MISSION SUPPORT SYSTEM (MSS) : GENERATING A TOOL CHAIN TUTORIAL FOR THE MSUI USER INTERFACE BY AUTOMATION OPERATIONS

About Me:

Name : Hrithik Kumar Verma
Github account : risehr
Slack/IRC Channel : Hrithik Verma
Timezone : GMT +5:30 (Indian Standard Time)

University Enrollment Details:

University Enrolled in : Cluster Innovation Centre, University of Delhi
Degree : Bachelor of Technology (B.Tech)
Major : Information Technology and Mathematical Innovations
Current Year : 3rd year
Expected Year of Graduation : 2022
Place : Delhi, India

Contact Details :

Primary Email ID : vermahrithik812@gmail.com

Why did I choose Python Software Foundation for GSoc'21 ?

Being an enthusiastic learner, I have always had my inclinations towards programming. I have collected experience in Java as an object oriented programming language since my high school. And that interest in object oriented programming and the broad experience that I gained with it led me to find Python much more suited towards my interests. Being an easy-to-learn and fun programming language, I started my Python journey and wrote small programs in it. Python really captures my attention due to the exciting things we can do with it which require a lot of time and effort in other languages.

Why did I choose MSS ?

The main reason why I chose MSS is the idea behind it. Since my childhood I have been a huge admirer of flights and space. The geographies of the world really excited my curiosity about its uniqueness. The first time I read the abstract of MSS, I was really hopeful and sure to know more about this and explore what was more inside of it.
Practically speaking, MSS is a very simply managed software with much abstraction. The approachable codebase that it has, cemented my decision to contribute to MSS, further. When I solved the first issue, I was out of my bounds and had since, always, loved to work towards it's development.

Contributions to MSS:

Contributions	Issue Number	Pull Request Number (merged)
Enhanced the UI of KML overlay, added color icons close to filename in the listWidget that shows color of all KML files with height of icon corresponding to linewidth, made the process faster. Added these features to KML overlay dockwidget.	#644	#709
On closing the main MSS window, the tableview with dockwidget was sticking around and not closing. I closed it by fixing the bug.	#736	#758

Project Details:

Organisation : **Python Software Foundation**
Sub-organisation : **Mission Support System (MSS)**

I. GENERATING A TOOL CHAIN FOR TUTORIALS OF THE MSS CLIENT

Project Abstract:

Mission Support System (or short MSS) is a software that is used for atmospheric research flight planning by scientists and meteorological researchers. It is a tool that uses various sets of forecast data in combination with a web map service to analyse the data and place the waypoints for planning research flights accordingly.

The present documentation of the Mission Support System client software covers the basic installation and offers a brief description of the use of the MSS client UI but lacks a vivid and lucid explanation of all the elements and features of the User Interface. The practical usage of the software has been lacking in the existing documentation. To enhance the user experience and to convey a deeper understanding about the practical usage of the software, I propose to create comprehensive and auto-generated video tutorials by a programming script that will be a helpful guide for the user.

Detailed Description of the Project:

The basic idea is to create a video tutorial. But manual recordings using the software would not be a very good idea since the MSS software keeps on updating. Sometimes, there are new features added or sometimes, the whole graphical user interface changes and in such situations the manual video guide would be outdated and of no use. So to be at par with changes, the video tutorial must be automated such that by updating it's scripts, the video tutorial and guide must also update with no hassle at all. It is just like updating the tests or like updating a feature.

To accomplish this task, the Graphical User Interface of the MSS must be operated in an automated fashion. Python provides a wide range of libraries for GUI automation and things like web drivers are being commonly used. GUI automation tools such as the PyAutoGui framework in python, sikuli , and pygui bot based in python are all excellent automation tools and frameworks to auto generate the use of the software. The tool sikuli comes with an IDE and the functions are written in Java based languages such as Jython, Javascript, JRuby, etc. So, in our case PyAutoGui and pygui bot will suit the best.

There will be six basic steps to develop the tutorials.

1. Automation of the MSS GUI by PyAutoGui, pygui bot, and pytest.
2. Write descriptions for the usage of the software according to automation using time modules for intervals.
3. Convert the descriptions using googletrans API or pydeepl API to the major languages of the world to be shown as subtitles.
4. Text to Speech conversion of the translated text using Google Text to Speech and Playsound module. For the time being, we would focus on spoken English and spoken German/Hindi as output speech converted from the descriptions.
5. Record the automation with Numpy python module to store the converted screen frame data into arrays and OpenCV for computation of the data and subsequent conversions to a video file.
6. Update the GUI with a tool bar option of Video Tutorials in the 'Help' section of the MSS main window. Then, in the drop down menu there will be user scenarios of what a user wants to do or a specific section of the software that needs to be explained to the user. The designing part will be done with PyQt5. The videos will be embedded with Youtube and a link will be joining the UI to the related videos in a modern design fashion.

Combining the six steps stated above, there can be 6 major tasks:

Task 1.

Automate all the windows of MSS and all the functionalities and features that it withholds. Using the modules stated in the above mentioned steps (in step 1), the automation process may take place. The following order does not reflect priority as to which window will be covered first. The main objective is to demonstrate that for the various use case scenarios that a user needs the MSS software for, these are the windows that will be covered in that operation and need to be automation updated. The windows are:

(i) ui_mainwindow
(ii) ui_mscolab_window
(iii) ui_topview_window
(iv) ui_sideview_window
(v) ui_tableview_window
(vi) ui_mscolab_admin_window
(vii) ui_mscolab_project_window
(viii) ui_mscolab_version_history
(ix) ui_add_user
(x) ui_add_project
(xi) ui_mscolab_merge_waypoints_dialog
(xii) ui_kmloverlay_dockwidget
(xiii) ui_remotesensing_dockwidget
(xiv) ui_satellite_dockwidget
(xv) ui_wms_dockwidget
(xvi) ui_wms_multilayers
(xvii) ui_wms_capabilities
(xviii) ui_wms_password_dialog

(xix) ui_topview_mapappearance (xx) ui_hexagon_dockwidget
(xxi) ui_sideview_options

The flowchart, for example, has been shown below in Fig.1 for the process of automation of one basic use case scenario of the MSS main window. After that all the circles can be extended in other groups of automation or independent operation can be made from other windows. Suppose the user wants to know what is there in the main MSS window for him. So, this flowchart can be used to explain to him the various tools, features that the window has and will also explain their subsequent functions to him in the video tutorial. Similarly other use case scenarios or operations can be generated such as “Explain the full working of Mscolab”, etc.

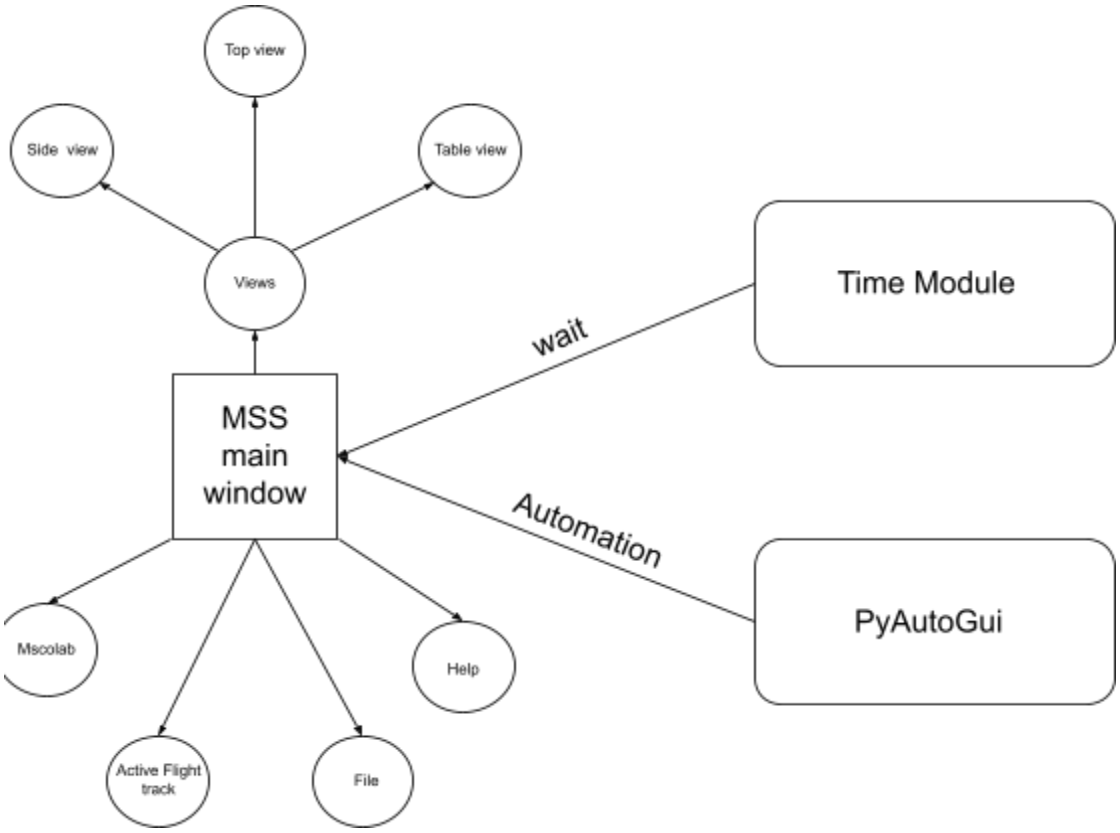

Fig 1 : The basic flowchart of one use case scenario among the various tutorial automation processes demonstrating the options available in the main MSS window.

Task 2.

Once all the windows or the use case scenarios are automated demonstrating and exploring various functionalities, a feature to show subtitles from written descriptions (which needs to be written in text files) along with the demonstration should be made through a defined function, *say def generate_subtitle(language)* in the preferred language of the user to be later integrated into the UI of tutorial video using the moviepy python library (See Task 5).

MoviePy is a very powerful library in python for video editing tasks such as concatenation, cutting, removing audio, inserting audio, inserting subtitles, etc. This task requires descriptions to be written about what is to be spoken while demonstrating the video and should be well-written and well-versed. It should clearly and very aptly describe the use case scenarios and related operations on the use of the MSS software. And no doubt a good delivery of english or any other language would be required. One should be good in content writing. The descriptions, primarily will be written in text files (.srt files) or simply in the python code separately, whichever will seem feasible and less time-consuming, during the time. These well-versed descriptions will be written in English first and then translated to any other language to be later generated as subtitles in the videos.

Task 3.

With all these things in place, a 'tutorial.py' script can be called, which takes continuously screenshots at 20 or 25 frames per second and converts it to a video with the specified duration given or after the specified user scenario demonstration is completed, all with the help of Numpy and OpenCV. The Numpy python module converts the taken screenshots into an array. The array will be in BGR format. The BGR is a 24-bit representation of the array(screenshot/image) where the lower-addressed 8 bits are blue, the next-addressed 8 are green and higher-addressed 8 are red. On the other hand, OpenCV converts this BGR format into RGB format and thus a sequence of these RGB formats, after some processing, generates the required video. And this generated video grouped for different use case scenarios can be saved in a python folder or directory folder with only the video component of the tutorial and later be uploaded to a remote server, such as Youtube, along with the audio and subtitles combined.

A pseudo code showing the live screen recording and thus, the above feature can be implemented in a similar fashion as follows:

```
# importing the required packages
import pyautogui
import cv2
import numpy as np

# Specify resolution
resolution = (1920, 1080)
# Specify video codec
codec = cv2.VideoWriter_fourcc(*"XVID")
# Specify name of Output file
filename = "Recording.avi"
# Specify frames rate. We can choose any value and experiment with it
fps = 25.0
# Creating a VideoWriter object
out = cv2.VideoWriter(filename, codec, fps, resolution)
# Create an Empty window
cv2.namedWindow("MSS Media Recording", cv2.WINDOW_NORMAL)
# Resize this window
cv2.resizeWindow("Live", 480, 270)
while True:
        # Take screenshot using PyAutoGUI
        img = pyautogui.screenshot()
        # Convert the screenshot to a numpy array
        frame = np.array(img)
        # Convert it from BGR(Blue, Green, Red) to
        # RGB(Red, Green, Blue)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        # Write it to the output file
        out.write(frame)
        # Optional: Display the recording screen
        cv2.imshow('MSS Media Recording', frame)
        # Stop recording when we press 'q'
        if cv2.waitKey(1) == ord('q'):
                break
# Release the Video writer
out.release()
# Destroy all windows
```

cv2.destroyAllWindows()

Task 4.

The most important part comes with the update of the UI of the main MSS window. There will be an option of “Video Tutorials” inside the already existing “Help” option of the main MSS window. Inside that option of Video Tutorials, a drop down menu will represent and showcase different use case scenarios of the MSS software for planning atmospheric research flights. When clicking on a use case or the very options denoting the use case, a link will be redirected to a browser window where tutorials videos would be played embedded inside a Youtube UI environment. One can also go to the main Youtube site and surf the videos or they can also come to the MSS official documentation page to play the videos. Here, the videos would be integrated with other relevant information also about how to use the software in flight planning.

Some of the use case scenarios might be listed as:

1. Explaining the User Interface of the main MSS window as to what options or functions it offers or to where it redirects.
2. Explain about the different views of an activated flight track and how to configure them.
3. Demonstrate the working of Mscolab or How one can collaborate with other people through MSS.
4. How to change configuration settings.
5. How to save a flight track.
6. How to import and export a flight track.
7. How to access the about or help documentation or the Tutorial Videos.
8. Explain KML Overlay dockwidget.
9. What is Remote Sensing in MSS and how to use it.
10. How to control Satellite Tracks.
11. What is Web Map Service?
12. What is Hexagon Control in Table View?
13. How to customize map settings and plotting behaviour.
14. How to add waypoints, move waypoints or delete waypoints.

These are some important use case scenarios which the user needs to understand. A detailed discussion will be with the mentors on what to include it in there apart from the very functioning of that particular use case. *These above listed points are subject to change.*

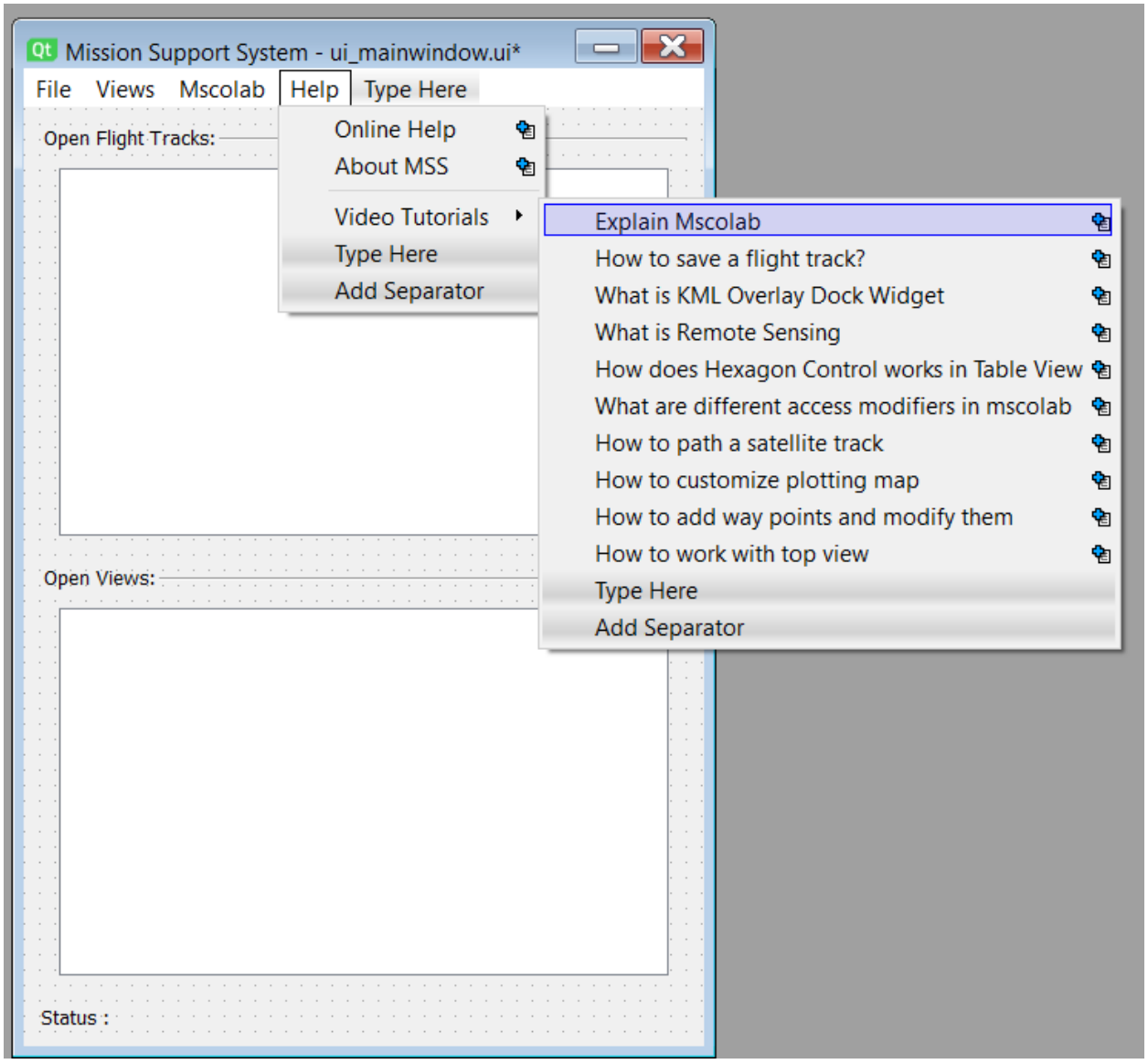

Fig 2 : The minor update in the UI. Listing all use case scenarios. Sections and Subsections. (subject to change)

Task 5:

The videos that were generated keeping the use case scenarios in mind need to be filled in with the audio and subtitles in proper synchronisation with the video. The subtitles can be converted to audio using the Google Text to Speech, and Playsound modules and then the audio must be integrated within the video with the help of the moviepy module of python. This task has to be programmed for the languages English and German/Hindi as the output audio. Next, these videos have to be uploaded to Youtube manually, choosing a variety of subtitle languages to give

the option of the captions being displayed in various languages of the world when the videos are played on Youtube.

For this, the comprehensive descriptions of the use cases already written in #Task 2, must be converted to other languages in which we are interested to show our subtitles in. The descriptions then need to be converted to different pre decided languages using the googletrans or pydeepl API and then we have to generate .srt files for particular languages and then we have to add these subtitle files with .srt extensions to Youtube, ultimately.
This task is done to cater to a wide international audience who will see the tutorial videos and be able to use the MSS software.

Task 6:

The uploaded videos into Youtube need to be embedded into the official documentation of MSS as well. The official documentation must also include the reference of this new feature (Generation of Video Tutorials in an automated fashion). The whole documentation needs to be updated and reviewed.
The online help system in the “Help” section, in this task, also needs to be adjusted.

Fig 3: A crude look of how the embedded Youtube video will look in the documentation. (subject to change)

II. Writing tests for the changed code and increasing its coverage.

The tests are an integral and important part of any large software foundation. It makes our life easier by allowing us to test the written or modified code before the user uses and tests it manually. The main goal should be to increase the coverage of the unit tests as much as possible to disregard any possibility of bugs and inconsistency in the code, at the base level at infancy, when the code is in the developing process. Therefore, detailed, accurate and quality unit tests should be deployed in the MSS software redesigning and tutorial development process and should have well-versed documentation for the same.

Timeline:

My proposed timeline is according to the GSOC 2021 timeline mentioned in the website. There are in total 10 weeks of coding.

Note:

I plan to work on all days in a week keeping the end of the week as a buffer period for any bug fixes, code refactoring and other code improvements and if I have to learn something that is needed in the following week.

At the end of every week, I will write the tests for the new code added or modified.

Time Span	Work
Upto May 16	❏ Work on more issues listed and to-be listed on Github/Open-MSS/MSS. ❏ Familiarize myself with the code base more efficiently along the process. ❏ Brush up all the necessary topics needed along with pytest and PEP style of coding.
May 17	**Student Projects are announced.**
May 17 - June 6	Community Bonding Period ❏ Communicating with the mentors about the best way to execute the project idea and which will be the best among the listed libraries, modules or APIs to work with. ❏ Finalize the project workflow and take suggestions on how to proceed.

	❏ Learn thoroughly about the various methods, modules that need to be implemented and have a deeper understanding of those topics. ❏ Communicate with other selected students and their projects to know how we can be of help to each other in a supporting environment.
June 7	**Official Coding Begins**
Week 1-2 June 7 - June 20	❏ Start to automate the Mission Support System GUI for some selected use case scenarios to better understand working with the module. Task #1 ❏ Brushing up the PyAutoGui library or whatever tools required for it.
Week 3-4 June 21 - July 4	❏ Wrapping around the Task#1 and covering every window to demonstrate the various *use case scenarios(1 - 14)* listed in Task#4. ❏ Discussing with the mentors about more use case scenarios and get a better understanding of the scientific terms. ❏ Since it is a new feature, tackling major issues and improvements.
Week 5 July 5 - July 11	❏ Automate more use case scenarios after critical discussion and analysis with mentors. ❏ Run the written automated scripts and see if the automation works perfectly in the required intervals or not. ❏ Writing code for recording the automation by taking multiple screenshots in required FPS and converting to a mp4 or preferred video file format, that is, Task#3
Evaluations **July 12 - July 16**	**Mentors and Students submit their evaluations of each other.**

Week 6 July 17 - July 23	❏ Task#2 : Writing “Descriptions” in text files and in clusters about the working flowchart of the automation video for all MSS windows. Basically it is the raw form of the to-be audio in the video tutorials. ❏ Maintaining the sync and deciding upon the timing intervals with the videos ❏ Writing code to convert these files into different languages for subtitles.
Week 7 July 24 - July 30	❏ Task#4 : Updating the UI to create MSS Tutorials option in the help section which will be linked to Youtube using PyQt5. ❏ Adding menu options to the “Help Section” to support the tutorials like listing the use case scenarios in a drop down menu.
Week 8 July 31 - Aug 6	❏ Task#5 : The most important feature of the project is to convert the ‘descriptions’ into audio and then integrate the audio into the video using MoviePy library. ❏ The subtitles earlier translated need to be converted to .srt files by defining the intervals and length of the video. ❏ To maintain proper synchronization of audio, video and subtitles will also be covered in this part.
Week 9 Aug 7 - Aug 13	❏ Wrapping the work around uploading videos to Youtube and linking them to the help section of the software. Task #5 ❏ Embedding Youtube videos in documentation. Task#6 ❏ Since the whole features of this part is complete, the code must be finally reviewed, refactored and bugs and issues must be addressed. ❏ Tests are written at the end of every week/slot mentioned in the timeline.

Aug 14 - Aug 16	❏ Writing more tests and increasing more of its coverage for the written code. (The tests would already be written at the end of each week for the work done in that week) ❏ Freeze code
Final Week **Week 10** **Aug 17 - Aug 23**	❏ Complete and update the old documentation and write about the new features and functionalities added in the MSS client. ❏ Fix bugs, issues, cover tests (if need arises in case) and prepare a final presentation of code to submit. ❏ Ensure no part is undone and everything completes smoothly. ❏ Final submission of code and documentation.
Final Evaluations **Aug 24 - Aug 30**	❏ Mentors review code submitted and determine that the student has passed successfully or not. ❏ Changes in the code requested by mentors should be addressed and then submitted again, if any.
August 31 2021	**Google Summer of Code 2021 Results are announced.**

Further Contributions:

I have a lot of ideas to work on for MSS even after GSoC 2021 ends and would love to stay a part of the community. I would always continue contributing to MSS in my spare time.

Projects such as Metpy Integration and Plotting Gallery Layer are important for learning and I would like to solve them later, if I get a chance.

I have previous works in the KML overlay section and I noticed that there is a lot that I can change. Starting from plotting of dotted, dashed and similar lines would be a nice feature I would like to add on. Placemarks and different types of routes and map files could be integrated. A 3D view of the map can also be generated to enhance flight planning.

The plotting of different types of layers, maps in flight planning can be done by cartopy and similar libraries of python and they have an important aspect and place in the software which I will try to build further.

Essentially speaking, MSS has a lot to develop and for budding developers like me, it will be a great opportunity and journey ahead.

Communication:

I will be available on the slack channel/IRC channel of MSS, mail and through calling during the working hours.
I will regularly be in touch with my mentors and update them constantly about the progress of my project and seek their help where I will be stuck.
I will be writing blog posts about the progress of my project and the things that I learnt.

Are you applying for other projects?

No, I am not applying for any other projects. Mission Support System is the only organization under the Python Software Foundation that I am applying for. I am sending only one proposal, that is this one, to GSoC.

Why am I best suited for it ?

I am a very passionate learner and I learn things easily. Because of my previous works, I have gained substantial knowledge about the technical skills required for this project that are Python, Git, PyQt5, UI design programming and various video editing and text-to-speech conversion APIs. Last year I had done an internship at CerebMedia as a Web Development Intern and UI designer also. Content writing was my hobby and in my spare time during the internship period, I have contributed content to the website. Most of the content you see in the website is my creation.
In my first year, I had an internship at Cluster Innovation Centre and there I contributed to a lot of projects on Python. Hence, I have enough experience required to fulfill the needs of this project.

I believe hard work and determination is something that can make you reach any goal and I am very consistent in that manner. Moreover, as I am in a learning phase, I am always ready to grasp anything and switch to anything easily that has a requirement for. I am willing to devote all my time and energy to this project and always ready to tackle problems and learn new things.

Other Commitments:

If I am selected for GSoc 2021, it would be my full time commitment during the summer except for the :

- ❏ End Semester Examinations of the University (15th May - 29th May Tentatively):
  - ➢ During this duration, time committed to GSoC will be 3-4 hours/day.
  - ➢ As the community bonding period will be on, I will make time for any meeting or conversations that are to be done.

I don't have any other commitments or summer plans during GSoC. In case something comes up, I will make sure to inform my mentor a week before and limit my unavailability to not more than 2 days.

Finally, I submit my Google Summer of Code 2021 proposal to Google and my mentors for their kind perusal.