A REPORT

ON

DEVELOPMENT OF A WEB APP WITH MAP BASED GUI FOR REAL TIME INLAND WATER LEVEL MONITORING USING SATELLITE ALTIMETRY

BY

<u>Names of students</u>	<u>ID. Nos.</u>
1. ABHIROOP BHATTACHARJEE	2016A3PS0702P
2. PRITHVI RAJ	2016A7PS0013P

AT

INDIAN INSTITUTE OF REMOTE SENSING, DEHRADUN

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
(JULY, 2018)

A REPORT

ON

DEVELOPMENT OF A WEB APP WITH MAP BASED GUI FOR REAL TIME INLAND WATER LEVEL MONITORING USING SATELLITE ALTIMETRY

BY

	Names of students	ID Nos.	Disciplines
1.	Abhiroop Bhattacharjee	2016A3PS0702P	Electrical & Electronics
2.	Prithvi Raj	2016A7PS0013P	Computer Science

Prepared in partial fulfillment of the

Practice School-I Course No. BITS F221

AT

INDIAN INSTITUTE OF REMOTE SENSING, DEHRADUN

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
(JULY, 2018)

Acknowledgements

The entire Practice School experience, has been a great learning experience. To all the people who made that possible, we extend our sincere gratitude and appreciation.

Firstly, we thank Dr. Prakash Chauhan, Director of IIRS, Dehradun for giving us this wonderful opportunity and providing us with all the necessary facilities.

We extend our most sincere gratitude to Dr. S.P. Aggarwal, Group Head of Water Resources Department, IIRS, for providing us with the opportunity to work on this project and for his valuable guidance at each and every step throughout.

We are also thankful to Mrs. Shefali Agarwal, Group Head of Geospatial Technology and Outreach Programme, IIRS, for her valuable support.

In addition, we wish to extend our gratitude to our PS Instructor, Dr. Chandra Shekhar, Associate Professor at BITS, Pilani – Pilani Campus, for his guidance, advice and support as and when needed.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)
Practice School Division

Station :	Indian Institute of Remote Sensing	**Centre :**	Dehradun
Duration:	54 days	**Date of start:**	22/05/2018

Date of Submission : 11/07/2018
Title of the Project : Development of a Web App with Map based GUI for real time Inland water level monitoring using satellite altimetry

Names/ID Nos./Disciplines of students :

1.	Abhiroop Bhattacharjee	2016A3PS0702P	B.E.(Hons.) Electrical and Electronics
2.	Prithvi Raj	2016A7PS0013P	B.E.(Hons.) Computer Science

Name of Supervisor : Dr. S.P. Aggarwal , Group Head of Water Resource Department
Name of Programme Coordinator : Mrs. Shefali Agarwal, Group Head of Geospatial Technology and Outreach Programme
Name of the PS Faculty : Dr. Chandra Shekhar

Keywords: Remote Sensing, Radar altimetry, Retracking Algorithm, Jason 2/3, Automation, django, netCDF, Maps API, Geo-visualisation
Project Areas: Altimetry and Automation

Abstract:

Radar altimetry data is now readily available in different formats, recorded by a large number of satellites. These altimetry datasets are available through a number of websites from data providers, satellite operators and manufacturers and the amount of data is huge. For procuring, processing and analysing such large datasets, currently there are only very cumbersome manual methods. The entire process of data procurement, processing and analysis has been automated and to give the user more flexibility, open-source tools have been used. A graphical user interface with map has been created that allows better geo-visualisation of the altimetry data. Open-source libraries with Python - a popular open-source language, have also been utilised for the said purpose.

Signatures of Students	**Signature of PS-I Faculty**
Date:	**Date:**

Table of Contents

1. Introduction:

1.1 Remote Sensing

Remote sensing is the acquisition of information about an object or phenomenon without making physical contact with the object and thus in contrast to on-site observation. Remote sensing is used in numerous fields, including geography, land surveying and most Earth Science disciplines (for example, hydrology, ecology, oceanography, glaciology, geology); it also has military, intelligence, commercial, economic, planning, and humanitarian applications.

In current usage, the term "remote sensing" generally refers to the use of satellite- or aircraft-based sensor technologies to detect and classify objects on Earth, including on the surface and in the atmosphere and oceans, based on propagated signals (e.g. electromagnetic radiation). It may be split into "active" remote sensing (i.e., when a signal is emitted by a satellite or aircraft and its reflection by the object is detected by the sensor) and "passive" remote sensing (i.e., when the reflection of sunlight is detected by the sensor)[1] (Fig : 1.1) **.**

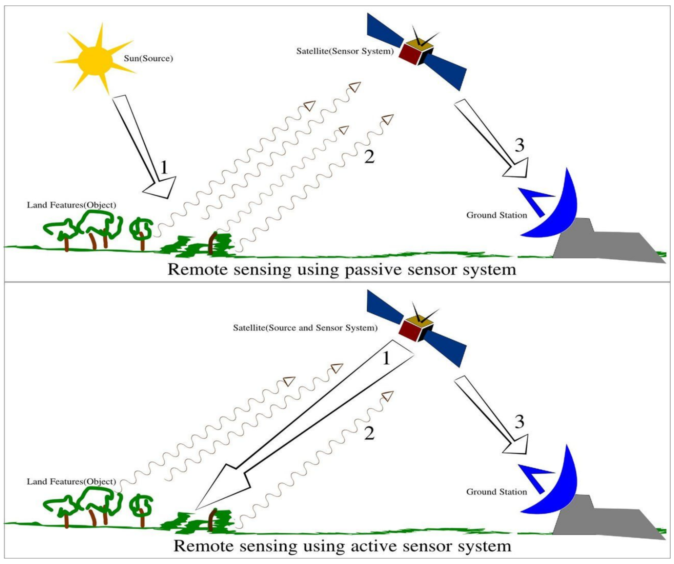

Sun(Source)
Satellite(Sensor System)
1
2
3
Land Features(Object)
Ground Station
Remote sensing using passive sensor system
Satellite(Source and Sensor System)
1
2
3
Land Features(Object)
Ground Station
Remote sensing using active sensor system

Fig: 1.1: Active and Passive Remote sensing

1.2 Altimetry

1.2.1 Basics of altimetry

Altimetry is a technique for measuring elevation. Satellite altimetry measures the time taken by a radar pulse to travel from the satellite antenna to the surface and back to the satellite receiver. Combined with precise satellite location data, altimetry measurements yield water-surface heights.

A lot of other information can be extracted from altimetry apart from altitude. The magnitude and shape of the echoes (or waveforms) also contain information about the characteristics of the surface which caused the reflection. The best results are obtained over the ocean, which is spatially homogeneous, and has a surface which conforms with known statistics. Surfaces which are not homogeneous, which contain discontinuities or significant slopes, such as some ice, rivers or land surfaces, make accurate interpretation more difficult.

Examples of altimetry satellites: Seasat, Geosat, TOPEX/Poseidon, ERS-1, ERS-2, Jason-2, Jason-3, Envisat, SARAL AltiKa etc. [2]

Several different frequencies are used for radar altimeters. Each frequency band has its advantages and disadvantages : sensitivity to atmospheric perturbations is more for Ku-band. Ka-band is better for observation of ice, rain, coastal zones and land masses.[3] Radar altimeters normally work in the E band, Ka band, or, for more advanced sea-level measurement, S band.

1.2.2 Altimetry waveforms

Consider a radar pulse emanating from a radar beacon traveling downwards and interacting with a flat ocean surface. Figures 1.1 and 1.2, show an illustration of the vertical cross-section and top-down view of the radar pulse.

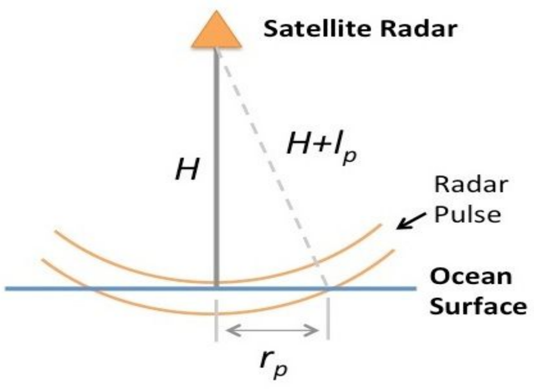
The radar altimeter measures the return power of the radar pulse that's reflected off the land/ocean surface. The temporal evolution of the reflected radar pulse is interpreted in order to estimate the distance between the radar altimeter and the reflecting surface; surface irregularities can also be estimated. The expected return pulse can be derived from a few basic mathematical considerations.

From the previous illustration of the radar pulse vertical cross-section, the radius of the outer edge can be found using the Pythagorean theorem:

$$\mathbf{H^2 + r_p^2 = (H + l_p)^2 = H^2 + l_p^2 + 2Hl_p}$$

where r_p is the leading edge of the pulse. If we assume l_p^2 is small and can be neglected then we can solve for r_p as:

$$\mathbf{r_p = (2Hl_p)^{1/2} = (2Hct_p)^{1/2}}$$



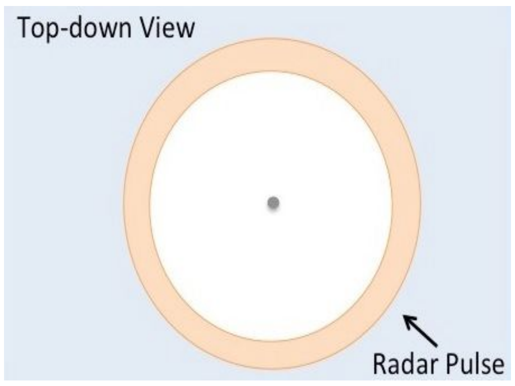


Fig: 1.2:Vertical cross-section of radar pulse striking ocean surface

Fig: 1.3: Top-down view of radar pulse

The time evolution of the return power measured from the footprint of the radar signal reflecting off the ocean/land surface can be described in three parts:

i. the time before the pulse arrives,
ii. the time after the beginning of the pulses arrival and before of the tail of the pulse has arrived,
iii. after the tail of the pulse has arrived.

The plot in Fig : 1.3 demonstrates the power function for a radar wave pulse. The power function should be constant with time after the full pulse reaches the ocean/land surface (at $(t-t_o)/t_p = 1$), however the power will actually decrease with time w.r.t the illumination pattern of the radar on the ocean/land surface.

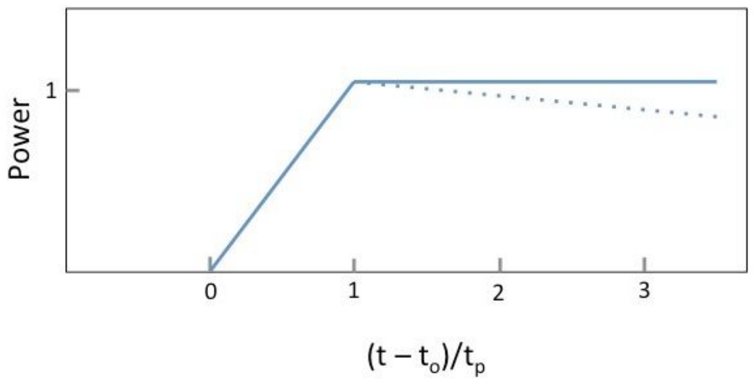


Fig : 1.4: Power function for a radar pulse

1.2.3 Sources of error

The following are the error sources associated with radar altimetry measurements:

a. **Tides** - Tidal variations are much larger than the dynamic variations in sea surface height. Because tidal periods can be on the order of diurnal and semidiurnal, the tides create an aliased frequency in the temporal variations in the sea level height that must be removed.

b. **Electromagnetic Bias** - There is a sea state bias where the troughs of waves tend to focus waves back to the radar, while the crests of the waves scatter waves away from the radar.

c. **Ionosphere** - The ionosphere can also impose a delay on the radar return signal, where electron plasma in the ionosphere slow down the group velocity of the radar pulse. The electron density in the ionosphere varies throughout the day, complicating the ionosphere correction.

d. **Dry Troposphere** - Refraction from the dry gas component of the atmosphere create a signal delay in the radar, but a correction can be approximated using the Saastamoinen formula: $\Delta R_{dry} = -0.02277 P_o * (1+0.0026\cos 2\varphi)$, where Po is the sea level pressure in Pascal and φ, the latitude.

e. **Wet Troposphere** - Water vapor can also cause a delay in the radar signal which can be more difficult to correct. A delay correction for the total water column in the radar measurement can be accounted for using output from meteorological models, like ECMWF and NCEP. [4]

1.2.4 Retracking

Retracking altimetry data is done by computing the departure of the altimeter waveform's leading edge from the altimeter tracking gate and correcting the satellite range measurement (and surface elevation) accordingly.

The major stages in the acquisition and tracking of the waveforms are as follows.

i. At regular intervals defined by the Pulse Repetition Frequency (PRF), frequency linearly modulated pulses are transmitted by the altimeter towards the Earth's surface.
ii. After reflection on the surface, the pulse is received back on board and mixed with a pulse similar to the emitted one which has been triggered by the tracker information.
iii. The mixed pulse, which is referred to as the 'individual echo', provides a sampled measurement of the return power as a function of time, distance or frequency.
iv. In order to reduce the statistical fluctuations (speckle) affecting the individual echoes and to perform real-time tracking (i.e. to maintain the signal inside an analysis window as far as range and power are concerned), these echoes are averaged on-board over a period which corresponds to the altimeter's duty cycle (typically 50 ms).
v. The resulting signal is referred to as an 'averaged echo' or a 'waveform'. It is processed by the on-board tracking system to derive the range and power.

The acquisition and tracking functions are carried out by two subsystems. The first one performs acquisition of the waveforms, this is the Radio Frequency Unit (RFU). The second one processes the waveforms, this is the Processing and Control Unit (PCU).
Over topographic surfaces, a radar altimeter's on-board tracking system is unable to maintain the echo waveform at the nominal tracking position in the filter bank, due to rapid range variations. This results in an error in the telemetered range known as tracker offset. Retracking is the term used to describe a group of non-linear ground processing estimation techniques which attempt to determine the tracker offset from the telemetered echoes, and

thereby estimate the range to the point of closest approach on the surface. Peaky echoes from sea ice cause range tracking jitter, which also results in tracker offset. [5]

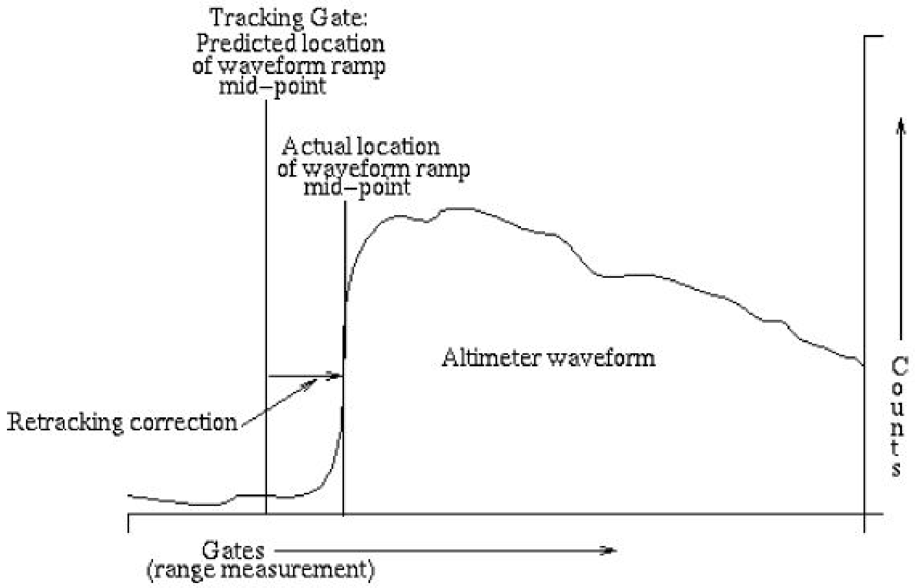

Tracking Gate: Predicted location of waveform ramp mid–point
Actual location of waveform ramp mid–point
Altimeter waveform
Retracking correction
Counts
Gates
(range measurement)

Fig : 1.5: Typical ice sheet altimeter waveform illustrating the retracking correction that must be applied to compensate for deviation of the waveform's leading edge from the on-board altimeter tracking gate. (Credits NASA/GSFC)

1.2.5 Formulae for calculating elevation

Finally, after retracking the altimeter waveforms and incorporating all other corrections vis-à-vis the error sources as mentioned earlier, the elevation of water level is calculated using the following formulae:

Elevation (from mean sea surface) = alt_20hz-ice_range_20hz_ku-(rad_wet_tropo_corr+model_dry_tropo_corr+pole_tide+iono_corr_gim_ku+solid_earth_tide)-mean_sea_surface

Elevation (w.r.t geoid) = alt_20hz-ice_range_20hz_ku-(rad_wet_tropo_corr+model_dry_tropo_corr+pole_tide+iono_corr_gim_ku+solid_earth_tide)-geoid

The names parameters on the R.H.S. of the above expressions are the names of variables that retrieved from the netCDF files that contain the satellite datasets.

2. Problem Statement and Objectives

Improving the flood forecasting capability of a downstream and highly flood prone areas in India is a fundamental hurdle . A ‘domestic’ flood- monitoring system over land covers only a small percentage of the total drainage area of the Ganges-Brahmaputra-Meghna river basins and is therefore able to produce skillful flood forecasts only up to 3 days of lead time. A week-long lead time is considered the minimum for effective decision making in the humid, seasonally-flood-prone environments for farmers, rural inhabitants and national governance.The disproportionate relationship between occurrence and impact of transboundary floods has also been attributed to the clear lack of institutional capacity for real-time communication of flood propagation status among upstream and downstream countries.

There is now growing body of work using satellite remote sensing to predict or forecast discharge. If satellites could provide an indirect way of measuring the upstream flow or water level, then the accuracy of a downstream forecasting system could be improved considerably. In this regard, satellite altimeters, that can provide real-time measurement of elevation of water bodies, are particularly effective. Satellite altimetry has progressed considerably over the last decade to become a viable alternative for many hydrological applications (e.g., Birkett, 1995; Schumann et al., 2009). [6]

2.1 Motivation

Real-time monitoring of water levels in inland water bodies requires inputs - remote sensing data
products collected by various satellites - in specific data formats. The output files generated are also in a particular format, so, processing and analysis of these datasets requires special tools and softwares. Doing all of this work manually or by using licensed softwares, which have several issues is difficult as well as cumbersome. The prime concern being the time taken for computation due to the large size of datasets for long-term data analysis. Another factor is that the licensed softwares being used are very expensive and have limitations. Users cannot make modifications as per their requirements, even rendering the softwares useless, in certain cases.
To optimise data processing and analysis, and to give the user more flexibility, open-source tools have been used and the entire process of data collection and processing so as to extract the necessary informations has been automated.

2.2 Objectives

a. Automating the process of-
    i. Checking for availability of new satellite data on the web
    ii. Downloading all new files available
    iii. Processing the netCDF files to get the elevation of required locations.
b. Determining longitudinal and latitudinal tolerance to be given so as to get points which lie only within the inland water body of choice.
c. Determining offset between the satellite’s actual ground track (from netCDF file) and offline ground track (from .kmz file).
d. Development of web app with map-based GUI for plotting output data of above process for different locations (water elevation VS time) to monitor variations.

3. Data Sets and Formats used

3.1 Data Sets

We have processed and analysed the following data products collected by various sensors on different satellites:

- **Jason 3 OGDR** [7] **:** The Jason-3 mission follows in the footsteps of the TOPEX / POSEIDON , Jason-1 and Jason-2 missions and provides continuity with those missions, with two driving ambitions:
  • Ensuring continuity of high quality measurements for ocean science
  • Providing operational products for assimilation and forecasting applications.

  OGDR stands for Operational Geophysical Data Record. This data set is regularly updated and a new file is available roughly every two hours (approximately the same time as one revolution of the satellite).
  This low latency is necessary since we wished to monitor the water level in real time for effective forecasting applications.

- **Jason 2 OGDR** [8]**:** This data set is very similar to Jason 3, the only difference being that the orbits are different. Use of two satellites with different orbits is advantageous as it will give more points of intersections with water bodies where we can use the satellite data to get the water level.

3.2 Data Formats

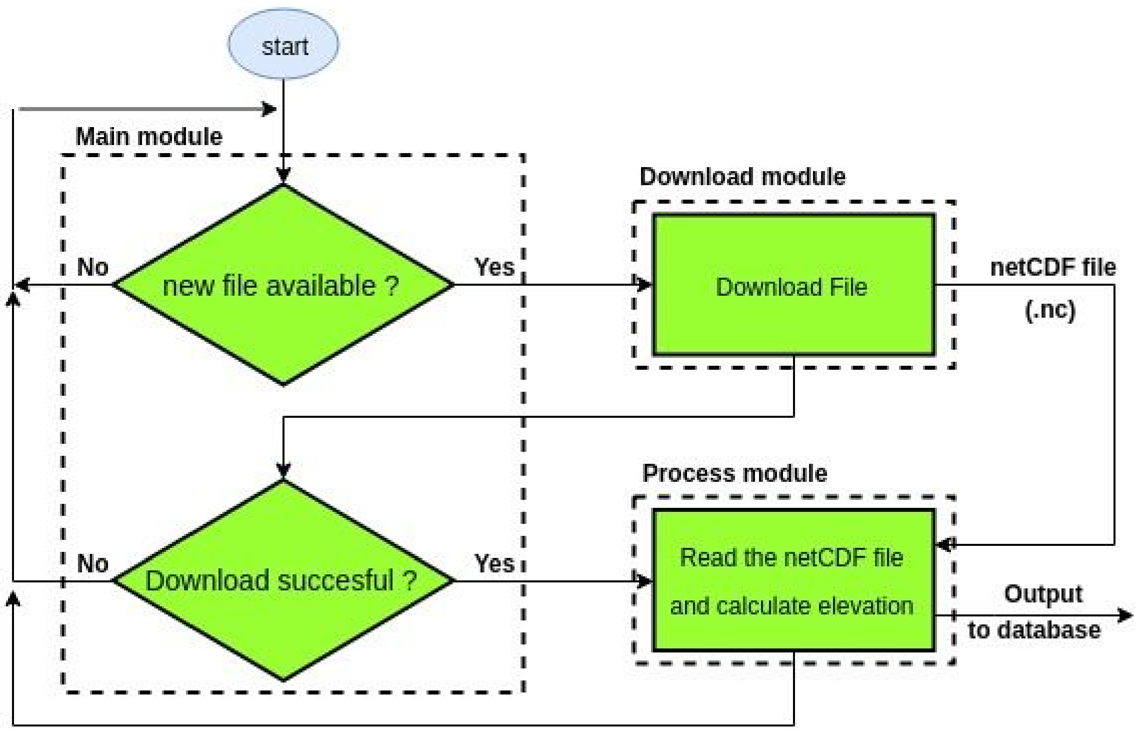
- **Text (.txt):** To store some simple log data for the python scripts involved in the automation process.

- **NetCDF (.nc) :** Network Data Common Format. It is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. We used the python netCDF4 library to process the netCDF files using python [9].

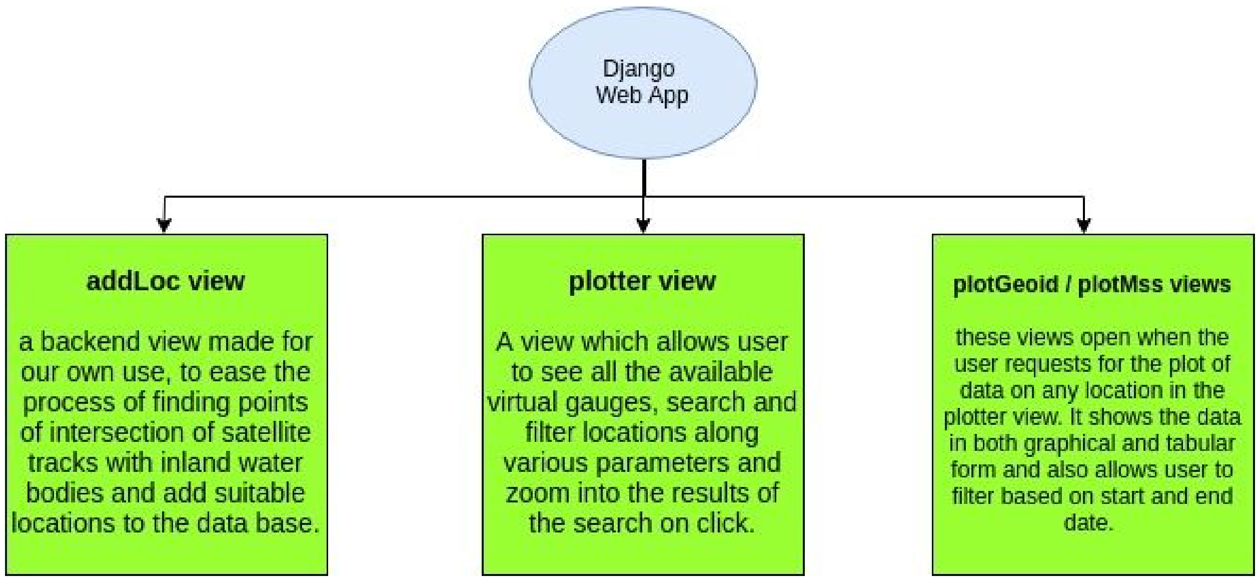
4. Methodology

4.1 Flow Chart

4.1.1 Automation Process



4.1.2 Django web app


Django
Web App
addLoc view
a backend view made for our own use, to ease the process of finding points of intersection of satellite tracks with inland water bodies and add suitable locations to the data base.
plotter view
A view which allows user to see all the available virtual gauges, search and filter locations along various parameters and zoom into the results of the search on click.
plotGeoid / plotMss views
these views open when the user requests for the plot of data on any location in the plotter view. It shows the data in both graphical and tabular form and also allows user to filter based on start and end date.

4.2 Languages and Libraries

Python - It is a widely used high-level programming language for general purpose Programming. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. [10]

Libraries used:

1. **django -** Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so one can focus on writing an app without needing to reinvent the wheel. It is free and open source.
   Django's primary goal is to ease the creation of complex, database-driven websites. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. [11]

2. **numpy** - It is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3. **os** - The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows the user to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

4. **netCDF4** - netcdf4-python is a Python interface to the netCDF C library. This module is used for reading NetCDF (.nc) files and extracting multi-dimensional data stored in these files.

5. **urllib3** - This module provides a high-level interface for fetching data across the World Wide Web. urllib is a package that collects several modules for working with URLs:
   - urllib.request for opening and reading URLs.
   - urllib.error containing the exceptions raised by urllib.request. [12]

HTML - Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. [13]

CSS - Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. [14]

JavaScript - JavaScript is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded. [15]
API used:

> **Bing Maps V8 control**- The Bing Maps V8 control is one of the most universal mapping controls available. Not only is it supported on standard PC & Mac browsers, but it is also supported on many mobile platforms. This maps API is ideal for web-based applications with support for JavaScript and TypeScript. The Bing Maps V8 Control SDK reduces development time by requiring less code to implement more functionality. Bing Maps V8 Control directly integrates content including administrative boundary data, a spatial math module and versioning support to simplify development. [16]

4.3 Directory structure of the software

The following is the basic top level directory structure of the software.

- App
    - Code
        - webGUI
            - Addloc
            - Media
            - Plotter
            - webGUI
            - __pycache__
    - Data
        - Last
        - Temp
        - JA3

        - JA2

4.4 Automation of downloading and processing of satellite data

The following set of programs is used to automate the process checking for a new file, downloading it and processing it to extract the elevations for required locations and add it to the database. We used python because of the ease of availability and use of it's various libraries for tasks like reading netCDF files, getting data from the web etc.

The following are the modules present in app/code/webGUI:

- **main.py** : this module checks for the existence of new files. It is basically a continuous loop which calls the jasonNewFileCheck( ) function.

    - **void jasonNewFileCheck(satName, ogdr_cycles_page_URL) :** this function takes as parameters the name of the jason satellite ("JA3" / "JA2") and also the URL to the page showing all the cycles of data available (defined as global variables).
      It then uses the urllib3 library to download the webpage and reads through the HTML file to find new undownloaded files. The downloaded web pages are stored in app/ data/ temp and they are overwritten every time so that unnecessary clutter is not created.
      The details of the last downloaded files are stored in app/ data/ last/ JAx_last.txt . this file contains the the last downloaded cycle and last downloaded file and whenever the main.py script is run, it checks for and downloads new files starting at this location. The last downloaded details are updated every time a new file if downloaded so that in case of any malfunction, on restarting the main.py script it will pick up where it left off.
    - **void checkData(satName, lcycle, ogdr_cycles_page_URL, lfile) :** This function takes one particular cycle and checks for new .nc file. If a file is found, it calls the download() function from download.py and once the file is downloaded it calls the process() function from process.py.

- **download.py :** this module only has one download function which downloads the given .nc file from the web page. The downloaded netCDF is present in app/ data/ temp/ ncf.nc. he file is overwritten every time so as to avoid unnecessary clutter.

- **process.py :** this module process the netCDF file and finds points close to the locations added to the django database. It reads locations from the database using django's database API. It also does the additional job of saving the files if the track passes through India (rectangular region with $65^{o}<$longitude$<100^{o}$ and $5^{o}<$latitude$<40^{o}$). Files are saved in app/data/JAx/cycleXYZ/ depending on the satellite and cycle number.

4.5 Determining longitudinal and latitudinal tolerance

The coordinates of the location present in the database at which water elevation is to be found may not be present in the datasets of the satellite in exact value. So, a tolerance has to be given so that the algorithm looks for a few points that are present in the vicinity of the specified point. We have considered a square neighbourhood around the location within which points in the datasets are selected and the elevation at these points is averaged to get the final elevation for the location. The square neighbourhood is filtered by selecting points fulfilling the condition given below:

|Latitude given - Latitude in dataset| <= ε **and** |Longitude given - Longitude in dataset| <= ε

Through trial and error, the value of **ε** has been suitably estimated to be 0.0045° such that the points in this neighbourhood are within roughly 600m of the desired location (clearly the waterbody chosen should be wide enough to accommodate such distances as the shortest distance between two measurements of the satellite is around 400m).

4.6 Determining offset between actual satellite track and .kmz file

While processing the netCDF files, it was observed that there was an offset between the coordinates of locations in these files and the satellite track traced by the .kmz file (Fig : 4.6.1) [17]. So, in order to quantify this variation, it was arbitrarily assumed that the longitudinal offset may vary with the altitude and/or the latitude of a location. The following data were obtained when the netCDF files for Jason 3 satellite (cycle 086) were processed (Table 4.6.1) :

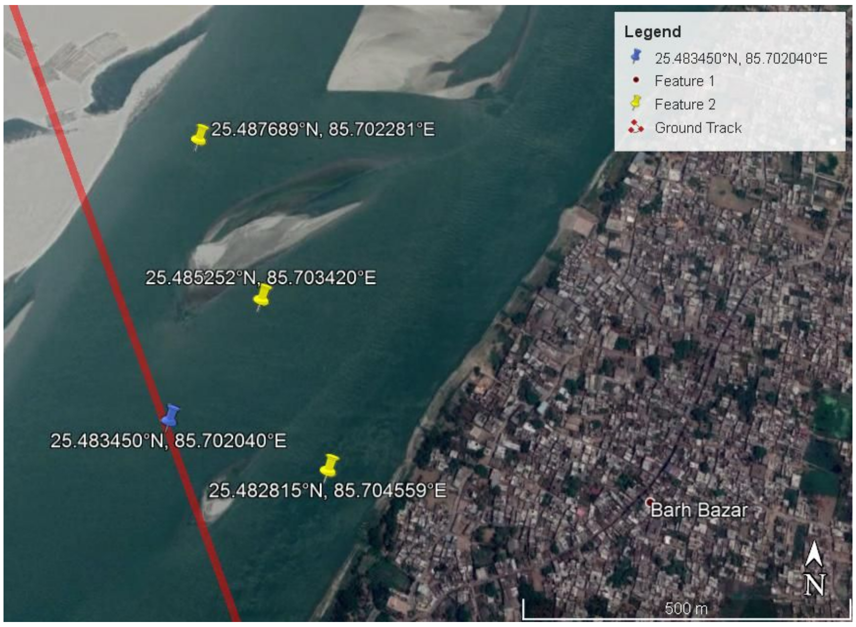


Fig : 4.6.1: Offset between satellite's original track and that in .kmz file

Table 4.6.1: Longitudinal offsets at different locations

	Track 53				
Elevation (m)	Track longitude	Latitude	Longitude	Offset in deg.	Mean offset
71	93.474991	26.748031	93.476673	0.001682	0.00169
	93.476532	26.750461	93.477841	0.001309	
	93.477578	26.752891	93.479009	0.001431	
	93.477839	26.75532	93.480177	0.002338	

	Track 53				
Elevation (m)	Track longitude	Latitude	Longitude	Offset in deg.	Mean offset
4272	97.818771	34.86705	97.819931	0.00116	0.00111
	97.820194	34.869418	97.821352	0.001158	
	97.821536	34.871785	97.822773	0.001237	
	97.823308	34.874153	97.824193	0.000885	

	Track 03				
Elevation (m)	Track longitude	Latitude	Longitude	Offset in deg.	Mean offset
483	80.623648	23.487756	80.625978	0.00233	0.002282
	80.624804	23.490203	80.627075	0.002271	
	80.625916	23.49265	80.628172	0.002256	
	80.626998	23.495097	80.629269	0.002271	

	Track 03				
Elevation (m)	Track longitude	Latitude	Longitude	Offset in deg.	Mean offset
4820	84.345531	31.103176	84.347756	0.002225	0.0023718
	84.346546	31.105577	84.349044	0.002498	
	84.347962	31.107977	84.350332	0.00237	
	84.349276	31.110377	84.35162	0.002344	
	84.350487	31.112777	84.352909	0.002422	

	Track 14				
Elevation (m)	Track longitude	Latitude	Longitude	Offset in deg.	Mean offset
-90	87.86601	20.620111	87.868047	0.002037	0.00205625
	87.867043	20.617652	87.869092	0.002049	
	87.868043	20.615192	87.870138	0.002095	
	87.869139	20.612732	87.871183	0.002044	

	Track 14				
Elevation (m)	Track longitude	Latitude	Longitude	Offset in deg.	Mean offset
-12	37.33024	63.912068	37.331673	0.001433	0.0014205
	37.335711	63.911005	37.337119	0.001408	

Based of the above data, the following graphs were plotted to figure out how the offset values vary with altitude and/or latitude of a location. Fig : 4.6.2 depicts the longitudinal offset values for various altitudes while Fig: 4.6.3 shows a plot between mean offset for each location with respect to altitude. Fig: 4.6.4 shows a plot between longitudinal offsets and latitudes for each location.

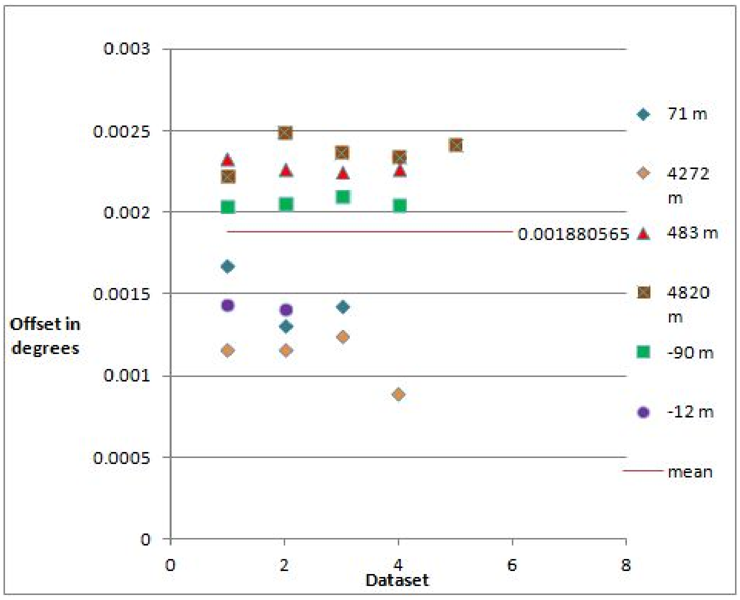


Fig : 4.6.2: Plot showing the various longitudinal offsets

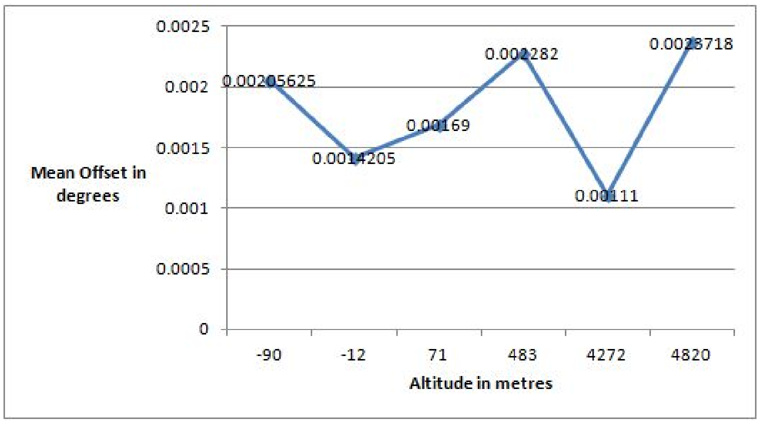


Fig : 4.6.3: Plot between mean longitudinal offset at each location and altitude

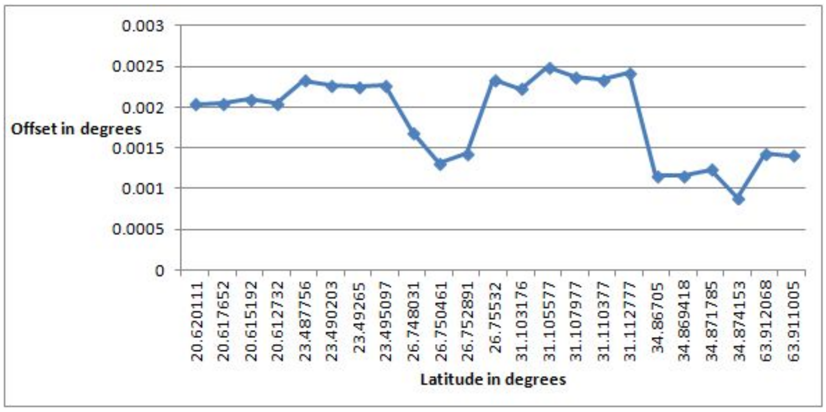


Fig : 4.6.4: Plot between longitudinal offset and latitude

From the above analysis, it was concluded that the variation of longitudinal offsets with latitude and altitude is arbitrary.

In the end, it was thought that the variation might be due to the gradual drift of satellite from its original position. On testing this theory it was found that there was drift, but the drift was also arbitrary , as Fig: 4.6.5 illustrates. Points obtained from cycles 007, 046 and 087 of Jason 3 satellite were plotted to get the following lines. As it can be seen, the variation with time is not in one direction.

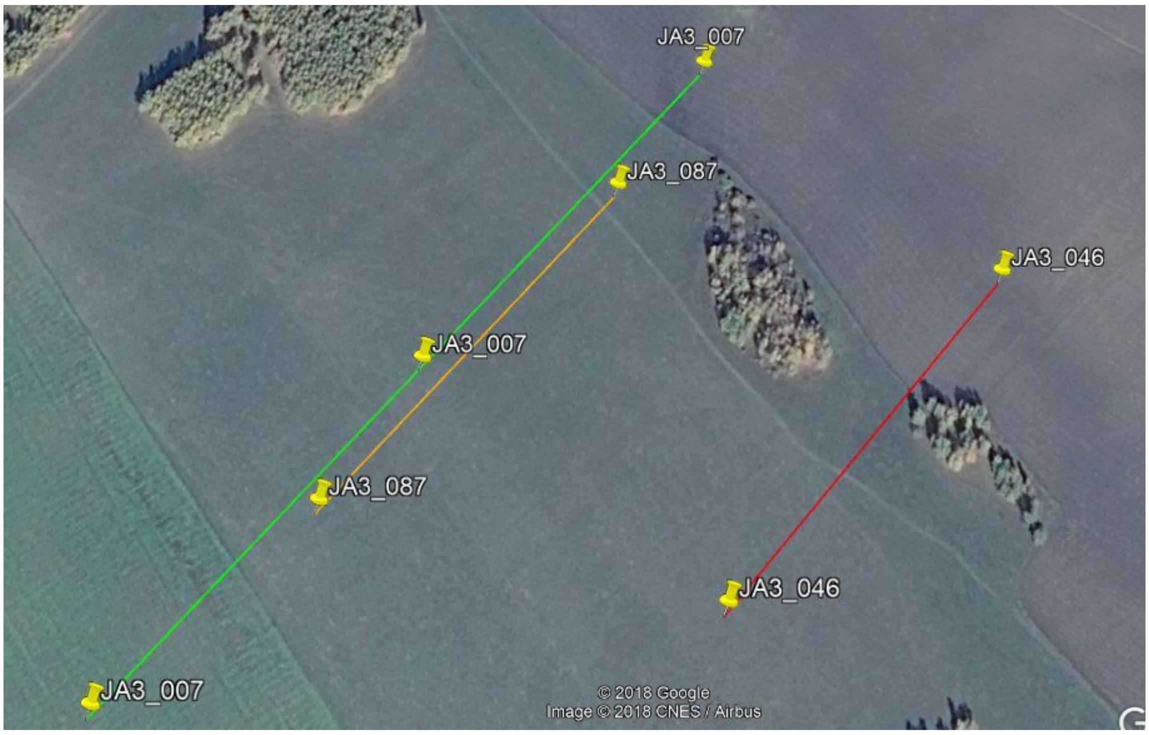


Fig : 4.6.5: Variation of satellite track with time

4.7 Web app with Map based GUI

The python Django web framework has been used for making this web app and integrating the GUI with the database. HTML, CSS and JavaScript have been used in making the front end of the website with considerable emphasis given on user experience. We have used the bing maps API for generating the maps and performing actions on it. It is free for all scientific and non-profit usage.

Django's database is manipulated by using classes called models defined in models.py file for each app (in our app, we have only defined models in addLoc/models.py). Models are analogous to tables in a conventional database. Django uses these model definitions to make tables in the database.

The following are the models present in our web app.

- **Satellite :** to store the different satellites being used
- **Country :** to store countries
- **State :** to store the states
- **WaterBodyType :** to store the type of water body

- **Basin :** to store the basins
- **Location :** to store information about different locations (has foreign keys referring to all the above models)
- **ElevationGeoid :** to store the elevation with geoid as reference. Has foreign key referring to location.
- **ElevationMss :** to store the elevation with mean sea surface as reference. Has foreign key referring to location.

A django web app works by tying URLs with certain functions defined in views.py of every app. When the URL is requested, the corresponded view function is called which will access the database to get any necessary data, render the corresponding html template for the view and finally return an http response along with a "context" which is a dictionary like object which ties variable names to objects so that data can be replaced on rendering the template by the template engine.

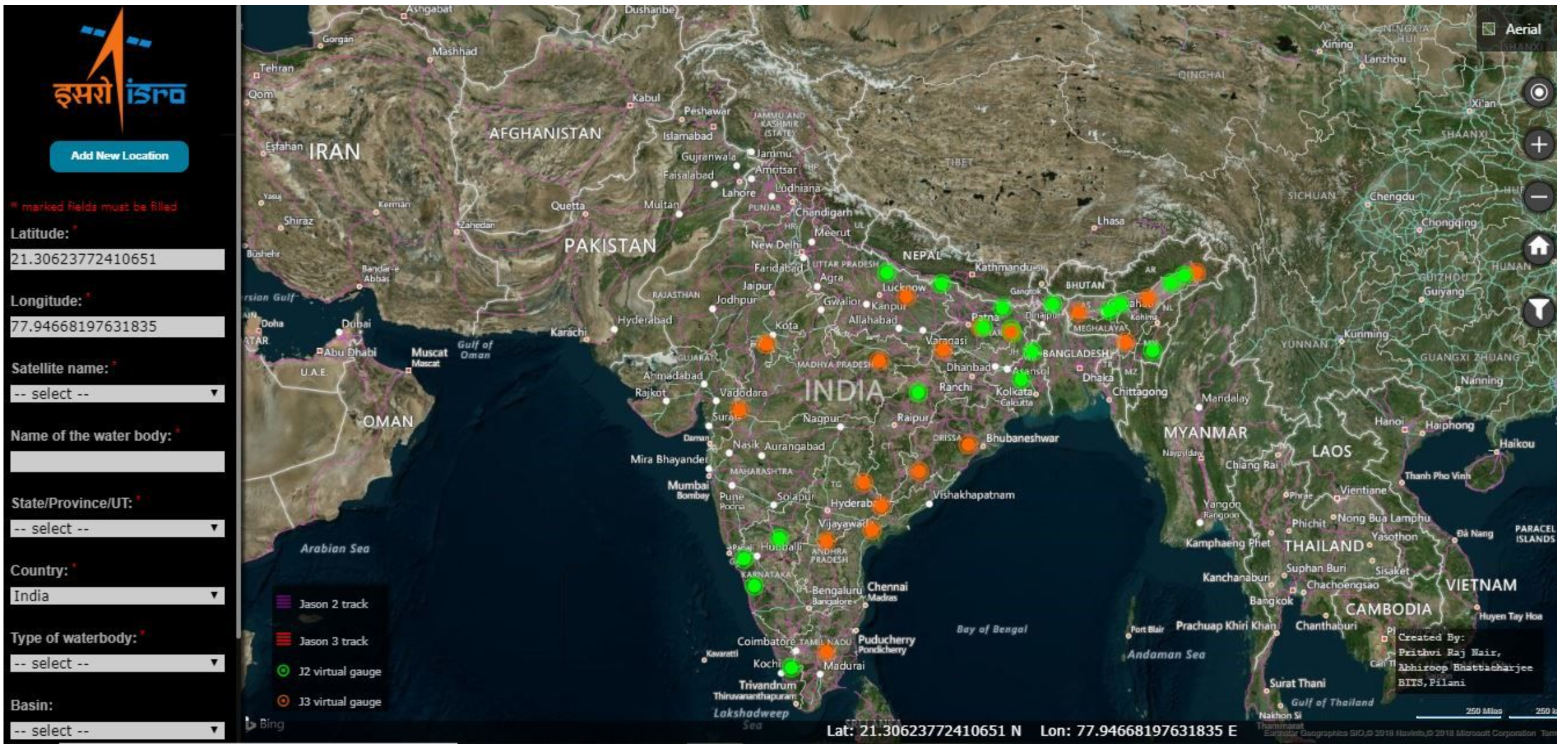
The following views are present in our app.

- **addLoc :** This is just a backend view (Fig : 5.1) we have made for our own use. It eases the process of finding locations where satellite tracks intersect the inland water bodies, and also provides a convenient interface to add locations to the database. This view is not meant to be publicly hosted.

- **plotter :** this is the main public view of the app (Fig : 5.2). This view allows user to see all the locations available on the map and also provides detailed info about each location when mouse hovers on it. There is also facility to search and filter locations along various parameters and zoom in to the search results. User can also view the satellite tracks overlayed on the map using a toggle button on the right (Fig : 5.4). Different map views are also offered, satellite and street views , which can be selected from the top right (Fig : 5.3).

- **plotGeoid / plotMss :** these views are just to plot the elevation data for the selected points (Fig : 5.5).

5. Final product screenshots



* marked fields must be filled

Latitude: * 21.30623772410651

Longitude: * 77.94668197631835

Satellite name: * -- select --

Name of the water body: *

State/Province/UT: * -- select --

Country: * India

Type of waterbody: * -- select --

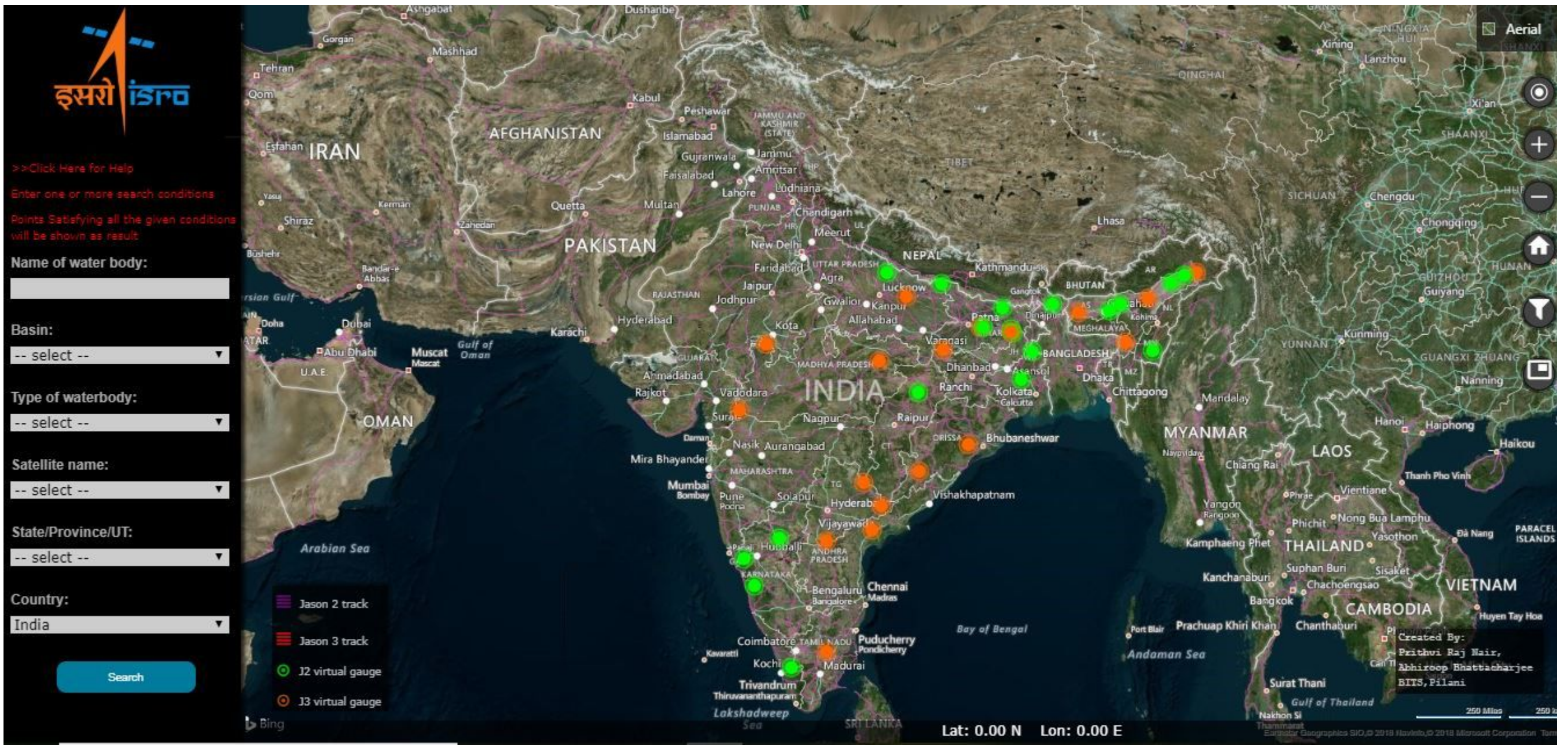
Basin: -- select --

▨ Jason 2 track
▨ Jason 3 track
● J2 virtual gauge
● J3 virtual gauge

Lat: 21.30623772410651 N Lon: 77.94668197631835 E

Created By:
Prithvi Raj Nair,
Abhiroop Bhattacharjee
BITS, Pilani

Fig: 5.1: addLoc view



>>Click Here for Help

Enter one or more search conditions

Points Satisfying all the given conditions will be shown as result

Name of water body:

Basin: -- select --

Type of waterbody: -- select --

Satellite name: -- select --

State/Province/UT: -- select --

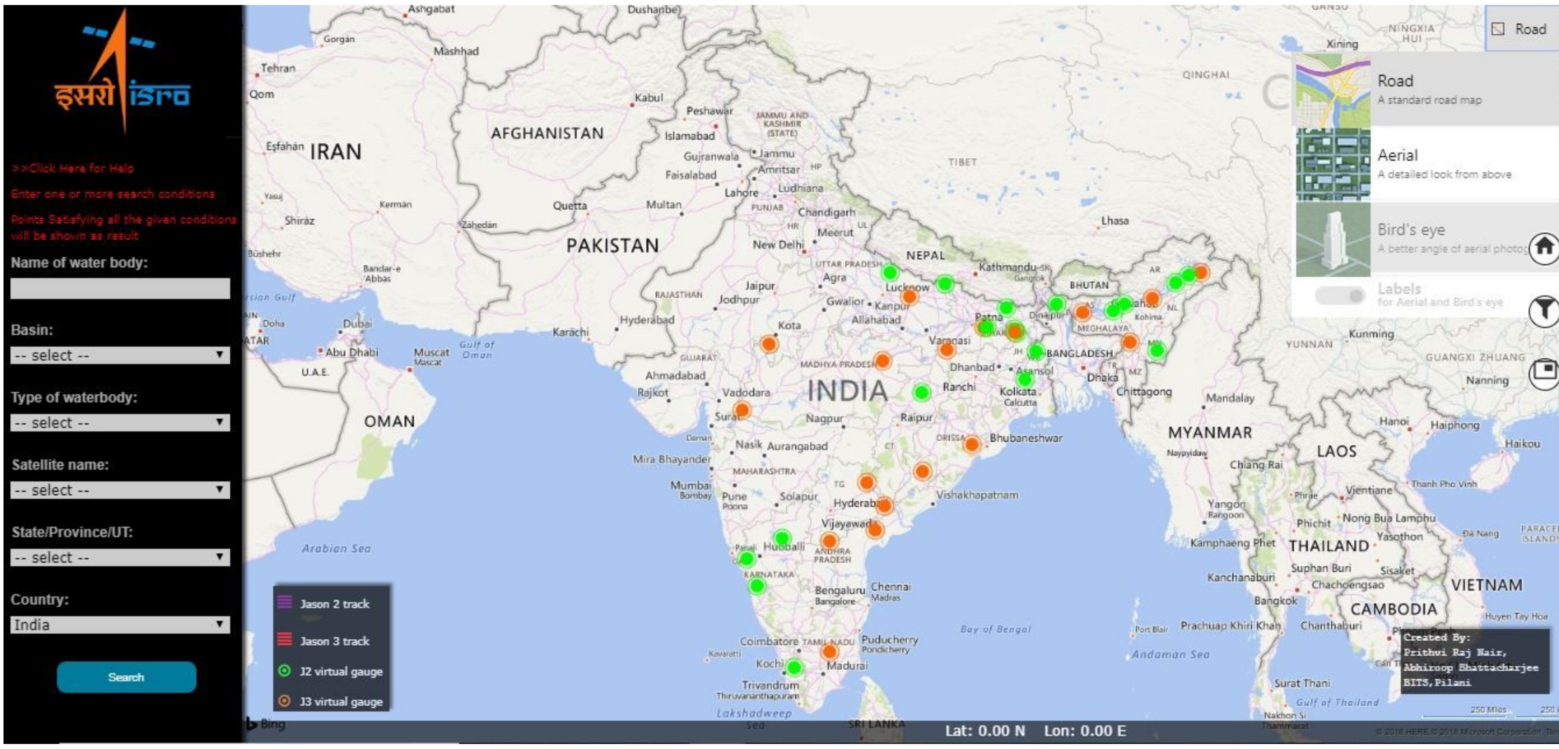
Country: India

▨ Jason 2 track
▨ Jason 3 track
● J2 virtual gauge
● J3 virtual gauge

Lat: 0.00 N Lon: 0.00 E

Created By:
Prithvi Raj Nair,
Abhiroop Bhattacharjee
BITS, Pilani

Fig: 5.2: plotter view



>>Click Here for Help

Enter one or more search conditions

Points Satisfying all the given conditions will be shown as result

Name of water body:

Basin: -- select --

Type of waterbody: -- select --

Satellite name: -- select --

State/Province/UT: -- select --

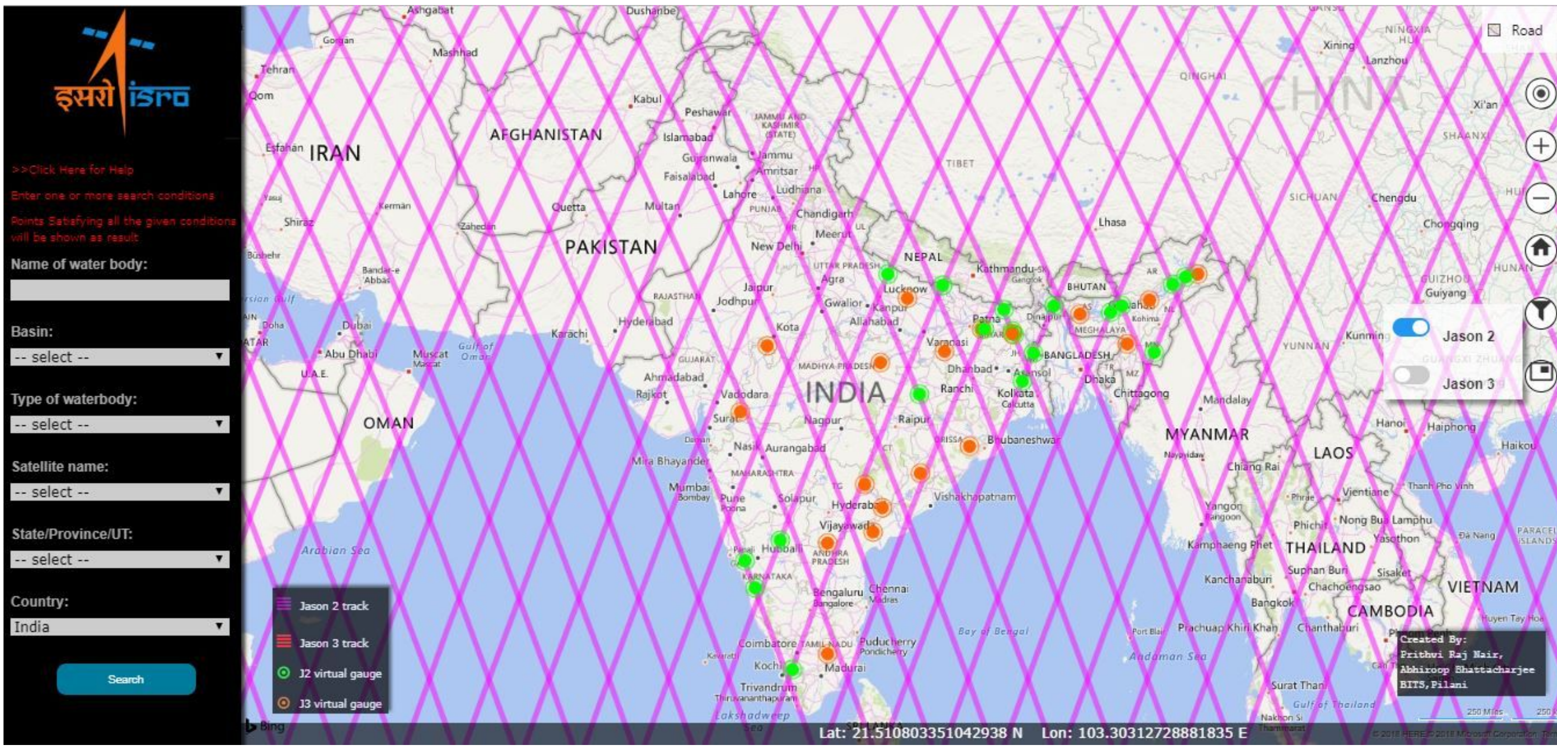
Country: India

Search

Created By: Prithvi Raj Nair, Abhiroop Bhattacharjee BITS, Pilani

Lat: 0.00 N Lon: 0.00 E

Fig: 5.3: map type selection



>>Click Here for Help

Enter one or more search conditions

Points Satisfying all the given conditions will be shown as result

Name of water body:

Basin: -- select --

Type of waterbody: -- select --

Satellite name: -- select --

State/Province/UT: -- select --

Country: India

Search

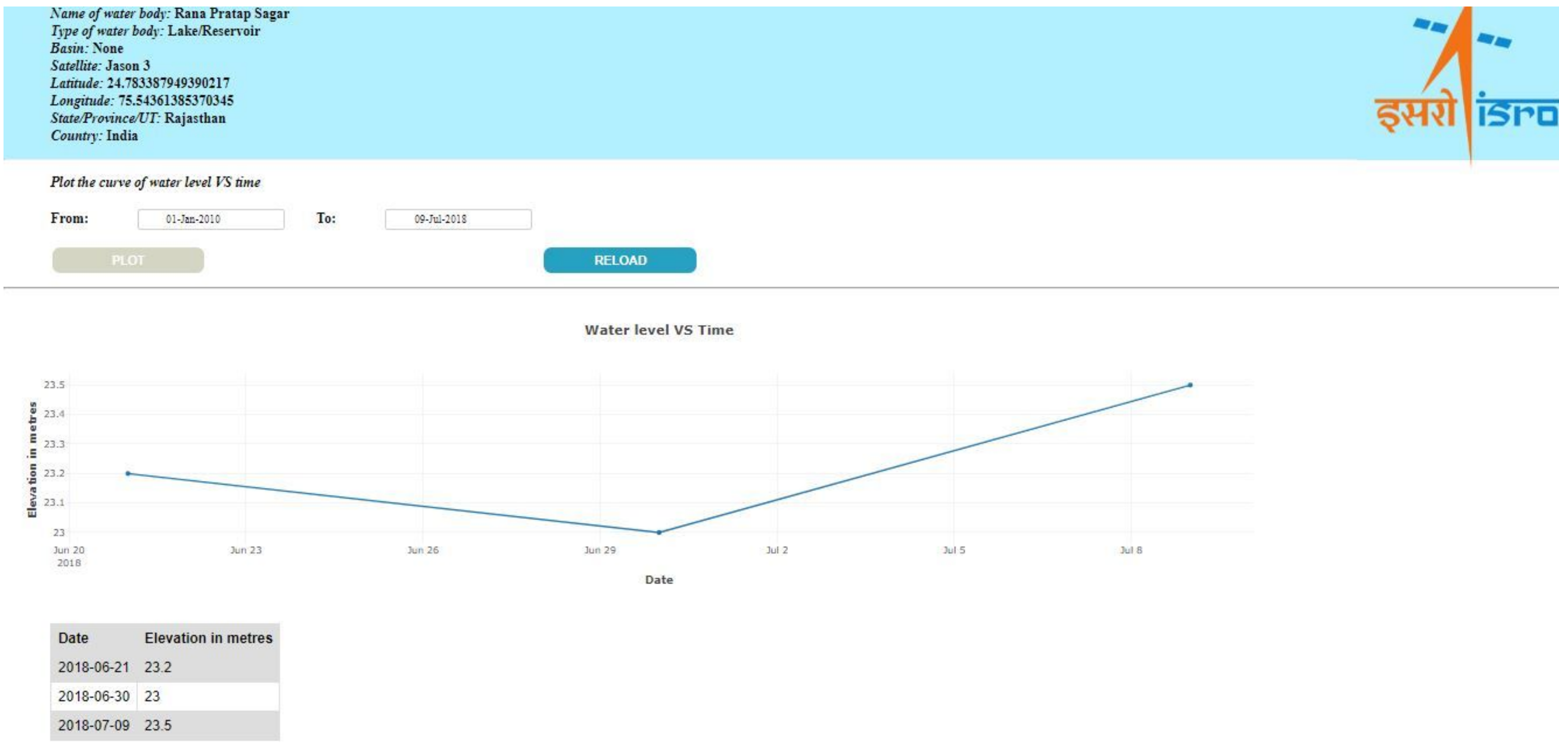
Jason 2 [on]

Jason 3 [off]

Created By: Prithvi Raj Nair, Abhiroop Bhattacharjee BITS, Pilani

Lat: 21.510803351042938 N Lon: 103.30312728881835 E

Fig: 5.4: Satellite track display on map


Name of water body: Rana Pratap Sagar
Type of water body: Lake/Reservoir
Basin: None
Satellite: Jason 3
Latitude: 24.783387949390217
Longitude: 75.54361385370345
State/Province/UT: Rajasthan
Country: India

Plot the curve of water level VS time

From:	01-Jan-2010	**To:**	09-Jul-2018

Date	Elevation in metres
2018-06-21	23.2
2018-06-30	23
2018-07-09	23.5

Fig: 5.5: Plot of data (sample data)

6. Conclusions and recommendations

6.1 Conclusions

- Most of the work that has been automated was earlier being done either manually which is very inconvenient.

- All the functions have been implemented using open source software.

- Better interface for the end user - The graphical user interface with map allows better geo-visualisation of the altimetry data. It is very easy to use and provides a variety of search and filter techniques. It improves user experience considerably when monitoring elevation of inland water bodies.

- The web app would allow effective monitoring of water levels at various locations and keep a watch as to when the water level goes abnormally high which could lead to floods. Real time plots (at most a few hours delay) are available at just one click and therefore it is a lot easier to monitor elevation on a frequent basis.

6.2 Recommendations

- Other satellite datasets (e.g. Sentinel 3) can be included in the system. Currently, it has not been done as the portal for downloading the data is highly complex and GUI based so automating the download is difficult.

- The database in use can be changed from Sqlite3 to faster ones like PostgreSQL. As the volume of data stored in the database at the current stage is small, so Sqlite3 works adequately well. For huge volumes of data (which will take a few years to accumulate), PostgreSQL or similar are recommended.

7. References

[1] https://en.wikipedia.org/wiki/Remote_sensing

[2] https://en.wikipedia.org/wiki/Satellite_geodesy#Altimetry

[3] https://www.aviso.altimetry.fr/en/techniques/altimetry.html

[4] https://en.wikipedia.org/wiki/Radar_altimeter

[5] *Rosmorduc, V., J. Benveniste, E. Bronner, S. Dinardo,O. Lauret, C. Maheu, M. Milagro, N. Picot, A. Ambrozio, R. Escolà, A. Garcia-Mondejar, M. Restano, E. Schrama, M. Terra-Homem, Radar Altimetry Tutorial; J. Benveniste and N. Picot (Eds),* http://www.altimetry.info, 2016

[6] *F. Hossain et al., "A Promising Radar Altimetry Satellite System for Operational Flood Forecasting in Flood-Prone Bangladesh," in IEEE Geoscience and Remote Sensing Magazine, vol. 2, no. 3, pp. 27-36, Sept. 2014. doi: 10.1109/MGRS.2014.2345414*

[7] https://data.nodc.noaa.gov/jason3/ogdr/ogdr/

[8] https://data.nodc.noaa.gov/jason2/ogdr/ogdr/

[9] http://unidata.github.io/netcdf4-python/

[10] https://en.wikipedia.org/wiki/Python_(programming_language)

[11] https://en.wikipedia.org/wiki/Django_(web_framework)

[12] https://docs.python.org/3.6/library/urllib.html

[13] https://en.wikipedia.org/wiki/HTML

[14] https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[15] https://en.wikipedia.org/wiki/JavaScript

[16] https://www.microsoft.com/en-us/maps/choose-your-bing-maps-api

[17] https://www.aviso.altimetry.fr/en/data/tools/pass-locator.html

[18] https://docs.djangoproject.com/en/2.0/

[19] https://msdn.microsoft.com/en-us/library/mt712556.aspx

Appendix

A. Documentation

A.1 Automation script

This part of the code is very well documented with comments and does not require any additional documentation. The description of the functions have been given above and there is nothing more to be added with regards to the automation script as it is quite simple.

A.2 Django web app

The django web app has many python files which it generates on it's own, like settings.py, manage.py etc. The functions of these files are vast and beyond the scope of the simple use of django that we have done. Documenting these files is not feasible as it will involve explaining the functioning of django as a whole which will be more than a 100 pages. If anyone wishes to know about django, it is best to read from django's official documentation [18]. Editing any of these files should be done with caution and should only be done once one is familiar with the functioning of django.

That being said, we have explained all the views and models in our web app above. For anyone wanting to understand how the web site and database are structured, that is more than sufficient.

B. Guidelines for running automation script

The following describes how to run the automation script on Dr. S.P. Aggarwal Sir's computer in his office in WRD.
For windows

1) open the folder `D:/Altimetry Tool project BITS Pilani (summer 2018)/app/code/webGUI` in command prompt
2) activate the virtual environment by using the following command -
   ```
   > conda activate altimetry_project
   ```
   OR
   ```
   > activate altimetry_project
   ```
3) run the main.py script using the following command-
   ```
   > python main.py
   ```

For Linux

1) Open folder `~/Altimetry Tool project BITS Pilani (summer 2018)/app` in the terminal
2) Activate the virtual environment
   ```
   > source env/bin/activate
   ```
   You will notice that the terminal prompt says "env" on the left
3) Change into the `code/webGUI` directory
4) Run the following command
   ```
   > python main.py
   ```

C. Guidelines for local hosting the web app

The following describes how to locally host the web app on Dr. S.P. Aggarwal Sir's computer in his office in WRD.

For windows

1) open the folder `D:/Altimetry Tool project BITS Pilani (summer 2018)/app/code/webGUI` in command prompt
2) activate the virtual environment by using the following command -
   ```
   > conda activate altimetry_project
   ```
   OR
   ```
   > activate altimetry_project
   ```
3) run the Django local host server
   ```
   > python manage.py runserver
   ```
   after running this command keep this terminal window open when you want to access the web app
   (press ctrl + C to stop the server)
4) open your browser, the URL for the views are as given below
   addLoc - http://127.0.0.1:8000/addLoc/
   plotter- http://127.0.0.1:8000/plotter/
   admin - http://127.0.0.1:8000/admin/

For Linux

1) open the folder `D:/Altimetry Tool project BITS Pilani (summer 2018)/app/code/webGUI` in command prompt
2) Activate the virtual environment
   ```
   > source env/bin/activate
   ```
   You will notice that the terminal prompt says "env" on the left
3) Change into the `code/webGUI` directory
4) run the Django local host server
   ```
   > python manage.py runserver
   ```
   after running this command keep this terminal window open when you want to access the web app
   (press ctrl + C to stop the server)
5) open your browser, the URL for the views are as given below
   addLoc - http://127.0.0.1:8000/addLoc/
   plotter- http://127.0.0.1:8000/plotter/
   admin - http://127.0.0.1:8000/admin/

ABOUT DJANGO ADMIN: django admin is just a page which allows you to manage the website database. It will show you all the tables present in the database and allow you to modify/add/delete data. (IMPORTANT NOTE: this view also shouldn't be hosted publicly)
The credentials for the admin super user for this web app are as given below
username: admin
password: adminpassword