ST. XAVIER'S COLLEGE [AOTONOMOUS], KOLKATA

B.Sc. Computer Science (Honours)

WISP: A PREFERENCE BASED LOCATION FINDER APPLICATION

by

Jimut Bahan Pal

Roll No: 508

Semester - VI

CIN – 16-300-4-02-0508

Under Guidance

of

Prof. Shalabh Agarwal

03.04.2019

ST. XAVIER'S COLLEGE [AUTONOMOUS], KOLKATA

WISP: A PREFERENCE BASED LOCATION FINDER APPLICATION

by
Jimut Bahan Pal
(Roll: 508)
Semester - VI
CIN – 16-300-4-02-0508

Under the guidance
of
Prof. Shalabh Agarwal

Submitted to the Department of Computer Science in partial
fulfilment of the requirements
for the degree of B.Sc.

St. Xavier's College, Kolkata
30 Mother Teresa Sarani
Kolkata – 700016
April 2019

CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled **Wisp: A Preference Based Location Finder Application** submitted to **St. Xavier's College** [Autonomous], Kolkata in partial fulfilment of the requirement for the award of the degree of **B.Sc. Computer Science (Honours),** is an authentic and original work carried out by **Mr. Jimut Bahan Pal** with roll number **508** under my guidance.

This project is a genuine work done by the student and has not been submitted to any other College / University / Institute for any purpose.

…….	...…….
Signature of the Student	Signature of the Guide
Date: ……..	Date: ……
Name and Address of the student	Name, Designation and Address of the Guide:
……..	……..
……..	……..
……..	……..
……..	……..

Roll No. **508**

ROLES AND RESPONSIBILITIES FORM

Name of the Project: **Wisp: A Preference Based Location Finder Application**

Date: 04/04/2019

Name of the Team Member	*Role	Tasks and Responsibilities
1.		

Name and Signature of the **Project Team members:**

Jimut Bahan Pal **Signature……..**

Signature of the Guide: …… **Date: …….**

* Students may take up roles such as Team Coordinator, Auditor/Receiver, Data Manager, Quality Manager or others according to the needs of the project.

ABSTRACT

The work of finding the best places according to user's preferences can be a tedious task. We designed, created and tested software which can take the preferences from the user and provide the best places according to their preferences from the location with a certain radius in real time. This eliminates the extra task of researching about a place and helps the user to visualise the places according to the preferences in real time. This uses foursquare API to fetch the data for the locations. We have built a console application for this purpose and this can be transferred to web based system in future. The internet connection is a must during using of this application. The Map that is formed is visualised in a browser and can be saved for future use. We have provided certain features in the application that will improve the user's experience.

ACKNOWLEDGEMENT

It is ritual that scholars express their gratitude to their supervisors. This acknowledgement is very special to me to express my deepest sense of gratitude and pay respect to my supervisor, Prof. Shalabh Agarwal, Head of the Department of Computer Science, for his constant encouragement, guidance, supervision, and support throughout the completion of my project. His close scrutiny, constructive criticism, and intellectual insight have immensely helped me in every stage of my work.

I'm grateful to my father, Dr. Jadab Kumar Pal, Deputy Chief Executive, Indian Statistical Institute, Kolkata for constantly motivating and supporting me to develop this documentation along with the application. I'm also thankful to Filipe Pires Alvarenga Fernandes, core developer, Physical Oceanographer, and writer of folium library in Python 3, for allowing me to use some part of their work for this documentation. I'm also thankful to XB Marketing for allowing me to use their work. Finally, I acknowledge the help received from all my friends and well-wishers whose constant motivation has promoted the completion of this project.

COPYRIGHT

PREFACE

In this documentation, the text material has been thoroughly revised. The Software Development Life Cycle of wisp application has been documented. The codes have been open-sourced and the references have been provided according to the materials. The original sources have been cited. I hope the documentation will help the researchers to maintain and develop the wisp application properly.

I'm grateful to my father, Dr. Jadab Kumar Pal, for his constant support to document and execute this project to completion. I'm also thankful to my guide, Prof. Shalabh Agarwal for supporting me at every stage of this project and providing constant motivation to finish the project. I also thank my mother Mrs. Sumita Pal, my brother Mr. Jisnoo Dev Pal for supporting and cheering up to complete this project. Finally, I'm thankful to all my well-wishers and professors of the Department of Computer Science for helping me with my career, and providing constant knowledge through all these three years of my B.Sc. degree, which has been invaluable to design this application.

FOREWARD

The author has always been keen to modify and develop the Wisp application software and document it by himself. He has made a sincere endeavour to bring the theory to application. I have noticed he has designed, developed, tested, and open-sourced the Wisp application by himself. He has provided full documentation about the software and credited the cited sources alike. He had worked a lot in the research and development of the wisp application and built the application from scratch all by himself.

I wish him all success.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background

While planning a tour to Pune in India, for the first time, we don't know the best place according to our preferences and benefits. We are unaware of the places that are present and we don't know which place is good for our stay. We may prefer to stay in a restaurant which is close to market, or we may prefer cold breeze of a river or a lake. Yet, we may also like a park near us during our stay, or maybe Cafe or Dominoes for those who like Pizza or Tea. We may also have some preferences or choices such as, maybe we want to find a place close to Museum or maybe some other choices. So, what is the most feasible way to find such places? One can search the internet about all the places that are near to their preferences and plot them in a Map. Then after plotting, they will take only those places which are densely clustered. Another way of researching is finding discussion forums about the places and then we can ask questions in the forums. But finding such forums is very difficult and most of the time even if we find any forums, the people are inactive. There is a difference in getting reply for a question in hours to getting reply in months, later for which the motivation to research dies. It may also happen that the people are not experienced enough and provide wrong information without verification. If one gets too lucky, they may even have genuine person living there. Since he/she might be a local person to the place, he/she might know every nook and corner of the city and may suggest some places according to the given preferences. So, they might give the location of a genuine place according to the given set of preferences, affordability, own rating etc. They might also have some knowledge about what other thinks about the place. So, the user might be very lucky to get all that quality information for deciding which place to go. In today's busy world, nobody cares about such phone calls and giving a long list according to preferences. Everybody is dependent on data and application which gives output in real time. Nobody have the energy and patience to call a friend and talk for hours just to get suggestions. They need answers in click of a button, that too with validation from a good number of people. So, we came up with an application named **Wisp [1].** The idea behind naming Wisp is it means a bunch of something in a standard dictionary, what it actually does is it takes the data points and clusters them in a bunch, so when we zoom the data points, it will resemble cross sectional wires.

What we have designed is a standalone application based on python programming language. It eliminates the problem for the user to do massive research works according to their preferences and helps them to find the best places according their needs in real time. It clusters the preferences of the users in a folium map. It also takes the data from an API known as Foursquare API **[2]**, which has active community to update the data very frequently and also it

is trustworthy API. There are two versions of the API, one is free, which is fine for one account of user for domestic uses and another is premium version for unlimited requests, generally designed for Organizational uses. The server for the API remains active for 24 hours a day, so it is very trustworthy and helpful for the user to use it any time. We build a full-fledged application for solving this problem of manually researching the places and we provided some extra features in our application. These extra features will help the users to use the application more effectively.

Some of the extra features are:-

1. There is wide variety of maps to select from.

2. We can visualise the output map in real time and also save it for later use in HTML file.

3. We have provided a secrets.txt file so that we don't have to manually insert the foursquare secrets and ID each time.

4. There are unlimited preferences that can be entered.

5. We can enter the location of almost any places and we can focus on those places only which forms a radius within that location. Both are inserted manually in the application.

6. We take the data from community maintained and rich API, called the foursquare API. We think that if we promote such API then there will be a lot of improvement in the actual data content of the API, which in turn will help in the data and performance of the application. More users will make an account and more amounts of data would be feed in terms of reviews comments etc. into the API.

1.2 Objectives

To develop an application, we need to make it visually appealing to the user and also we need to make it user friendly. Hence, we build Wisp, which can take parameters in text box and provide valuable results according to the preferences. There are parameters which the user can supply in the text box, such as location of the place which might be a city name, radius in meters from the city or location, number of preferences that the user wants to search upon. It also wants the Foursquare Client ID and Foursquare Client Secret as input which can be provided manually each time or may be given by a text file in the current directory. We provide a button which has the capability to read the text file containing the foursquare secret and ID as JSON data, so that we don't have to manually insert them each time. It takes the number of preferences from the user and creates a sub form accordingly, and the data is inserted and we can provide the type of the Map to be visualised on as an additional input. We can also provide an optional parameter for the name of the Map that will be saved as HTML file after real time visualisation. The data is fetched in JSON format, cleaned and plotted accordingly to the Folium Map. We can select a range of map present in the Folium **[3]** library in Python 3. We can select from Mapbox Bright, Stamen Toner, Stamen Terrain, Open Street Map and Mapbox Control Room. After the data is cleaned and plotted, it is visualised in real time using a custom made live HTTP server in Python3. We can even save this map for future use in HTML file, when we zoom in the HTML generated, we will see the clusters broken into sub-clusters to locate the places according to our preferences.

1.3 Purpose

The main purpose of this project was to visualise the location given by the users according to the preferences provided and display the map in real time. We found that there was no existing software that could actually help us to visualise the location in real time. The main aim was to create an application which is user friendly and takes the necessary arguments from the user and display the map in real time. We were motivated by the Foursquare API and the ease with which we can parse the data using the JSON library in python 3. With the help of a request, it will send all the data that are necessary to visualise the place in real time. It sends all the details of the place along with the latitude, longitude, name, rating etc. which helps us to cluster it in a map. We can focus

on a single place by providing only the radius for which we will find the preference.

1.4 Scope

The scope of the project is to make an application which is easy to use and understand; which is free of cost and shall help a wide range of users. This was created for the sole purpose to help the intended audience to know about a place. The only valuable item here is Data. So if the data is present, it will help to plot the place and visualise the real time Map of the place according to the preferences. There are two types of API access available to the users, one is free which when verified using the developer account, can make 10,000 requests per day. The premium account offers tips, menus, ratings, URL etc. for the users. We are using the regular API that suffices our needs to plot the data point according to the latitude and longitude in a map.

1.5 Applicability

There are the following quality attributes that the Wisp application provides for all the users:

- Availability – the foursquare API is a good API, maintained by a community of trusted users which verify and cross check each data. So the data is genuine and is available for 24 hrs a day, since there is a rate limit for regular users, a user can make about 10000 requests per day, which is too much for any user.

- Correctness – the foursquare API is used by almost 150,000 developers, so the data needs to be correct else foursquare wouldn't have been so much popular.

- Maintainability – since we have used modular design for the building and creation of Wisp, it is easily upgradable and maintainable. It will be open sourced and will help a lot of developers to collaborate and give it a new life.

- Usability – the Wisp application is user friendly. The UI is self-intuitive. Wisp also comes with a help option which gives the extra arguments that can be passed to wisp in command line for more features.

1.6 Achievements

Wisp has been successfully built in about 20 days' time period by following the design paradigms formulated by software developers and researchers. It is made using modular design and techniques and is open sourced **[1].** The application is user friendly and has received a lot of praise from the beta users. There has been about 8 cycles involved in the development of the Wisp software. The application works very well for almost all the locations and gives output in real time. The software has been put on Github, an open source community, where we hope it will take a new form in the future. More and more collaborators would help in designing of the software. After the tedious hours of debugging and analysing code for bugs, it is quite achieving to see an application like Wisp being able to deliver output on the fly to the users.

1.7 Organisation of the report

There are five parts to this report. The introduction has been already covered before this part. We will see the surveys of technologies preceding this section, where we will discuss about all the tools and planning involved in the creation of this project. We will see why we have chosen Python 3 over other programming tools. We will see the planning of the system, the design of documentations. The chart, diagrams etc. in the surveys of technologies. We

will also see some of the libraries in Python 3 through which this application was build. After the surveys of technologies, this project discusses about Requirements and Analysis of the planning. We will look in details about the planning of the building of the software. We will see how the GUI was build, we will see how the Command Line Interface was build, and how the argument parser was build and how the custom made HTTP server was build. We will look at all the details and diagrams of the development of the application. PERT and Gantt charts are presented in this phase. After that the System design phase has been discussed. We will look at the code, the modules, and the application in working state through screenshots. We will look at the GUI, the themes, the debug mode, etc. in this section. Last we discuss about the testing phase and the integration phase, where all the modules are coupled and made to an application. Then we will look at the future scope of this application and conclude this documentation.

CHAPTER 2: SURVEY OF TECHNOLOGIES

2.1 Preliminary Investigation

This is the first phase **[4]** in the design of the software development phase as shown in **Figure 1**. The main objectives of this system are to find out the needs of the system and basic investigation on how to build the system.

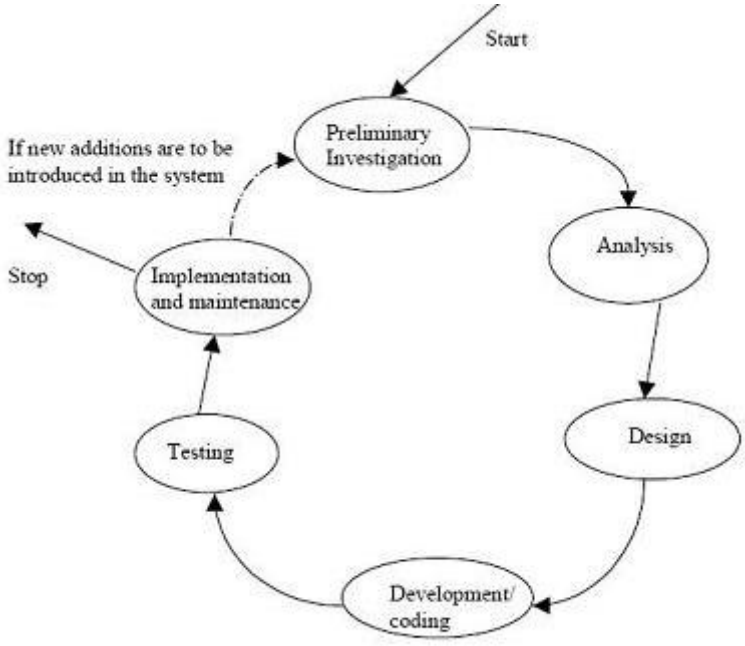

Figure 1: The design of the software development phase

In our case **[5],** we investigated the current scenario of the applications present, and we found that there were relatively different applications present. So, we investigated the needs of the users, the limitations present in the technology and finally designed a SRS according to the needs. Once the first investigation is done, all the alternative solutions are written and noted down. All these alternative systems generated are known as candidate systems. All the candidate systems are weighted and the best system is selected, that is known as "proposed system". Then the proposed system is evaluated for its feasibility. The feasibility of a system means whether it is beneficial to build the system or not. The feasibility is investigated from developer's and customer's point of view. The developers check whether it is feasible to build the system with

existing technology or not, and the customer checks their resources to afford the system and whether it is beneficial for them to have the system or not. The feasibility system is evaluated on three main issues: technical, economical and operational.

Technical feasibility focuses on the fact whether the development of the proposed system can be completed with the help of existing tools and equipment or not. Whether the developers are available, who can build such system, whether there are skills present in the existing team or not. It is generally done with the help of a project manager. Economic feasibility researches on whether there will be any benefits to the customer who are paying for the system? Is the system really different from other existing system? Is this system really user friendly? All these questions are answered in economic feasibility. Legal feasibility checks whether there is any legal hassle in developing the system, whether they have to pay royalty fee, whether any copyright infringement occurs later, whether they are not including any political parties or not, etc. Then comes the operational feasibility phase, which checks whether there will be any hassle from the users in using the system, whether they can provide update without damaging the existing tools present in the system, how much maintainable the existing software system is, etc.

2.2 Project Planning

Planning is in invaluable in creation of software **[6]**. It refers to everything that is done from the beginning to the end of the project. It is defining all the process that we go through during the creation of the software. During the initial phase the system requirements and the needs of the software are unknown, so it is necessary to plan for the right thing. This helps to give the project a correct path in development of the software. It helps to create landmarks and objectives during the creation of the project. The activities in project planning are actually varied at each step, since the dependencies, requirements and the objective that is achieved or to be achieved by the software actually changes over time. Most of the time in planning involves the need to get things done and what structure the developers should actually follow to get things done.

One of the good practices during project planning is to do brainstorming. We generally follow brainstorming during the development of the software. The main objective of brainstorming is to come up with ideas and keywords and write it in the board. The brainstorming generally occurs as a meet up with refreshments, and all the people present in the development team comes up. There are no rules for ideas to pop up, and all the new ideas are acknowledged.

This helps them to motivate and share ideas among themselves. Those ideas are noted down for future, and the project manager assigns the tasks according to those ideas. It is not only used in creation of new ideas, but also to make some fixes in bugs and all the innovation that is required in the development process of the creation of the project. Project management is a procedural step in the project management, where the required documentation is created to ensure successful project completion **[7]**. The documentation or plan in a project defines all the necessary documents that are required to integrate, prepare, define and coordinate additional plans. The project plan clearly defines the necessary responsibilities that are administered by the project managers, the role of the key developers, the role of the project leader, the role of the artist, etc. It is a blueprint that is required to help the team to refer it in future. The inputs to the project planning phase generally includes project schedules, meeting schedules, conceptual proposals, resource limitations, requirements and specifications of the tools that are to be used, success metrics, etc. It generally evolves over time, it starts with the scope and basic requirements of the first prototype and then it moves to deadline, the additional functions that are needed to be achieved etc. It depends on the tasks, deadlines, and additional checkpoints that are needed to be done to complete the project.

The planning is integrated into Gantt charts and Pert charts along with other types of charts. It provides a basic blueprint or roadmap to be followed by the developers present in the project. It gives a project overview for all the parties involved in the project.

The main steps that are always required by the project planning step are: -

- Road maps that are required to be completed during the project.
- Work required for completing the project.
- The key responsibilities and roles of all the people involved in the project.
- Minimum time for the project completion.
- Work required for the project completion.
- Major project deliverables.
- Milestones in completing the project.

Planning is never finished until the project is completed. It changes over time and the new dependencies and requirements arise at every step. Even the need of the software is unclear during the development of the project. This happens because generally the people who are clients are unaware of the

specifications and what exactly their software will need to achieve. This is because of the communication gap that, the people involved in the industry generally don't understand the naive terms and the customer too don't understand the actual need that is required to be done for the project. The project plan may return to the planning stage multiple times prior to the project completion, the complexity determines the length of the project planning stage.

CHAPTER 3: REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

Our aim is to create an application which will eliminate the task of researching a place according to the preferences of the user and would give the best places according to the preferences in real time. We came up with the idea of creating the application using Python 3 language, which will use the help of data and will plot the places in real time for the user.

When working in an organization, the ideas can come from the top **[8]**, i.e., the management team whereas it can also come up from bottom up, i.e., from us; in the case of our project we used both the ideas from our guide, i.e., top down and from us and it helped us a lot to develop our project. Identifying potential problems in designing the software can save a lot of time and money for a big project; hence, the identification of need is an important step in designing of software development. Problem analysis and planning is one of the most critical stages in the development of the software lifecycle, since the rest of the project will be based on the blueprint of this plan created in this step. If the project advances without the completion of this step, the project may go in wrong direction and may take longer time to develop and design.

The needs are identified after someone with expertise makes observation about the project. A good observer can answer all the needs of the projects by efficient observations. Therefore, it is necessary for an experienced person to tell the needs and requirements of the project by careful observations. The observation should meet all the needs of the project. The more the key questions are answered, the better the observation is. There are two processes involved in observations and gathering information. Observation highlights what is needed, whereas gathering information focuses on the process needed to execute the ongoing project. These two will actually include comment from the group that will benefit from the project.

3.2 Requirements Specification

A Software Requirements and Specifications (SRS) is the description of the full developed system that is undertaken as a project **[9]**. It lays out both the functional and non-functional requirements and also includes the set of used cases that describe the user interactions that the software must provide.

In order to fully understand a project that is to be undertaken, one needs to come up with a SRS document. It is needed by the people who will sponsor the project; it will help the project managers assign tasks according to the document; it may also help to raise funds during the execution of the project. The SRS have all the details of how to execute the project to completion. It also forms a blueprint for the developers to look at milestones and landmarks and to analyse the actual time that is required for the completion of the project. It helps to find on the limitations and the needs at the beginning of the project. It helps to find out the actual development time of the project. It helps the developers to give report whether they are lagging behind the SRS or are going ahead of estimated project time completion **[10]**.

The SRS that we had designed at the beginning of our project:

3.2.1 Introduction

3.2.1.1 Purpose

The main purpose of this project was to visualise the location given by the users according to the preferences provided and display the map in real time. We found that there was no existing software that could actually help us to visualise the location in real time. The main aim was to create an application which is user friendly and takes the necessary arguments from the user and display the map in real time. We

were motivated by the Foursquare API and the ease with which we can parse the data using the JSON library in python 3. With the help of a request, it will send all the data that are necessary to visualise the place in real time. It sends all the details of the place along with the latitude, longitude, name, rating etc. which helps us to cluster it in a map. We can focus on a single place by providing only the radius for which we will find the preference.

3.2.1.2 Document Conventions

The document uses the following conventions:

- API – Application Programming Interface
- JSON – JavaScript Object Notation
- GUI – Graphical User Interface
- PERT – Program Evaluation and Review Technique
- RAD – Rapid Application Development
- ID – Identity Document
- OS – Operating System
- UI – User Interface
- SDLC – Software Development Life Cycle

3.2.1.3 Intended Audience

The intended audience for this project are the common users that will benefit from this application. The people who wants to know about a place or those who wants to plan a trip about a place according to their preferences.

3.2.1.4 Project Scope

The scope of the project is to make an application which is easy to use and understand; which is free of cost and shall help a wide range of users. This was created for the sole purpose to help the intended audience to know about a place. The only valuable item here is Data. So if the data is present, it will help to plot the place and visualise the real time Map of the place according to the preferences. There are two types of API access available to the users, one is free which when verified using the developer account, can make 10,000 requests per day. The premium account offers tips, menus, ratings, URL etc. for the users. We are using the regular API that suffices our needs to plot the data point according to the latitude and longitude in a map.

3.3.1.5 References

1. https://developer.foursquare

2. Coursera, IBM data science specialisation Course.

3.2.2 Overall Description

3.2.2.1 Product Perspective

It takes the following inputs from the user:

- CLIENT ID – this is the client ID of the foursquare client, and helps to identify whether it is a premium account or a regular account. This is needed by the foursquare API to make requests and send the data to the respective users. This ID is done because of the fact that there is a record of the data that is collected

by the user, it helps to distinguish users and preferences, so that the server don't give premium content to the regular users or etc. This is done for security purposes by the server and the organization that manages the foursquare API. It also helps to calculate the number of calls that are sent by the user and rate limit the users for future. This is different for different users and is provided by the API vendor on successful creation of an account.

- FOURSQUARE SECRET – this is same like the client ID, but different from it by the fact that, it provides the token for the request. It has the actual secret. This is also different for different users, and helps to distinguish between the users. These helps the server to rate the limit for the users so that there are no unnecessary overloads in the server. It is done to ensure proper use of the API and data. These two data can be manually inserted every time or can be stored in a secrets.txt file from the folder which the wisp application is used. If we use the secrets.txt file, then we can use the use default secrets button to fetch this data and use it for future use.

- Location/City – this takes the location for which the user wants to find the details, according to their given preferences.

- Radius – this takes the radius in metres from the location in which the user will visualise the data.

- No. Of Preferences – this takes the number of preferences to visualise on so that it creates a sub-form and the user can enter the data accordingly.

- Map Type – this is the choice of Map that the user can select to visualise. It have five different types of map choices, they are Mapbox Bright, Stamen Toner, Stamen Terrain, Open Street Map and Mapbox Control Room. These are the inbuilt map types present in folium, and we can use it for visualisation

- Save map as – this is an optional argument that is passed to the GUI, which takes the name of the map to be saved in HTML for future references.

3.2.2.2 Product Features

This is a simple application which doesn't use the database for its backend. This is a real time visualisation application, which takes the data from the foursquare API. The data that is present in the foursquare API is of remote origin. This sends requests according to the number of preferences present, and gets back the JSON data. The JSON data is then cleaned using Pandas library and then converted to clusters using Numpy. After the forming of clusters, this is plotted using folium map. All these requests, cleaning of the data are done on-the-fly.

3.2.2.3 Operating Environment

This application is built in Python 3. Python 3 is an interpreted language, for this reason this is O.S. independent. We have built this application in Manjaro-Linux.

The details of the OS are shown in **Figure 2**. We can clearly see from the screenshot that it is built in Manjaro 18.0.4 Illyria O.S. Additionally the Kernel of the OS is x86_64 Linux origin. The other software and hardware requirements are given.

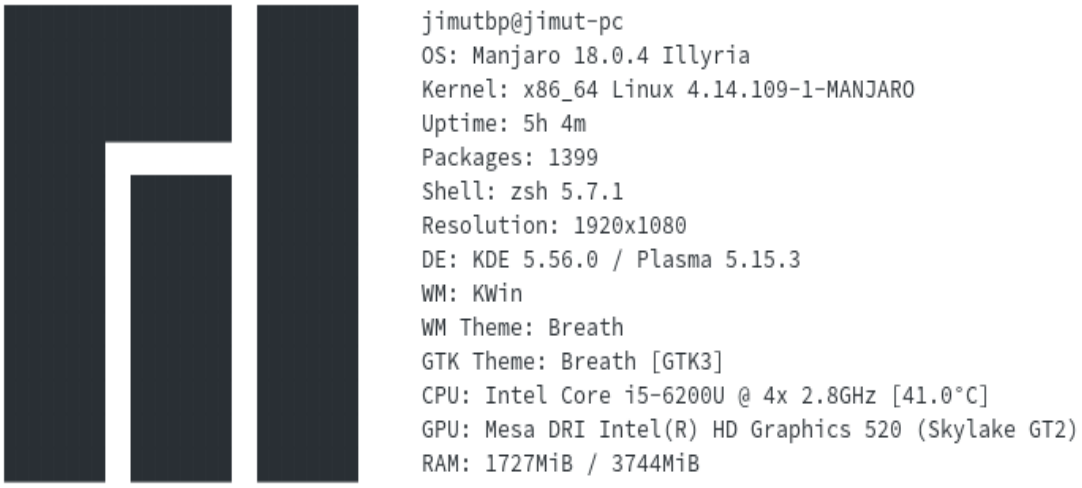

Figure 2: The Operational Environment in which Wisp was built

3.2.2.4 Design and implementation Constraints

The software should be build which solely serves the purpose of real time visualisation of the preferences of the places given by the users. The software building should follow the SRS document and shall have to be written in Python3 programming language.

3.2.2.5 Assumptions and Dependencies

The initial dependencies that were required were:

- requests==2.21.0
- pandas==0.24.1
- folium==0.7.0

- geopy==1.18.1
- numpy==1.16.1
- wget==3.2
- datetime (latest inbuilt version)
- IPython (latest inbuilt version)
- pip (latest inbuilt version)

We used the Python3.7.2[GCC 8.2.1 20181127] version for our project. The code was developed in the Manjaro OS and is built in such a way that it runs on all the OS, namely Mac, Linux and Windows. The only way that can be done is to use this command **pip install –r requirements.txt** which will install all the requirements and dependencies that are present in the text file named requirements. The help that is got from this command is Python provides a recursive way to install the dependencies from the text file present. This helps to install multiple dependencies in automation, without manually typing each dependency to be installed.

3.2.3 System Features

3.2.3.1 Functional Requirements

We believe in the power and development of open source tools, technology and software. For this reason, we chose to use Python3 as a language for coding this application from scratch. The reason behind choosing python3 over other languages is, inter-alia, it can do rapid prototyping of almost any software from scratch. We can build anything from it in less time. It has a community of developers which provide support in every field. It also has too many libraries for making things much easier for everybody. Python is rich in its Machine learning libraries, cloud computing libraries, web development libraries, computer vision libraries etc. The code written in python runs on almost all three OS, i.e., Linux, Mac and Windows, provided the developer uses the common libraries, so transferring this software from one OS to another is not a problem in Python3. Since our application needs a GUI for user to input the raw data,

we need a good data transfer library, a library for fast computation for doing machine learning stuffs and finally a browser to display the map. Python have all the libraries required for this project. We used here tkinter, json, numpy, pandas, ipython and folium as libraries to build this application.

3.2.4 External Interface Requirements

3.2.4.1 User interfaces

We have used tkinter to create the GUI of the application. Using tkinter, one could easily make custom GUI for almost any application. The tkinter library is so much flexible that it runs on all three OS. We have provided many parameters for the tkinter GUI, to create a visually appealing GUI. The software and the hardware interfaces that are used for testing the Wisp application are given.

We tested this software on a machine with the following specifications:

Architecture: x86_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

Address sizes: 39 bits physical, 48 bits virtual

CPU(s): 4

On-line CPU(s) list: 0-3

Thread(s) per core: 2

Core(s) per socket: 2

Socket(s): 1

NUMA node(s): 1

Vendor ID: GenuineIntel

CPU family: 6

Model: 78

Model name: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz

Stepping: 3

CPU MHz: 500.002

CPU max MHz: 2800.0000

CPU min MHz: 400.0000

BogoMIPS: 4801.00

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 3072K

NUMA node0 CPU(s): 0-3

Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp flush_l1d

3.2.4.2 Communication Interfaces

This project supports almost all the web browser present. The map that is generated is visualised in real time using a web browser. The visualisation that is created can be saved as a HTML file that can be used in the future for visualisation and can be opened only using a web browser.

3.2.5 Non Functional Requirements

3.2.5.1 Performance Requirements

The Wisp application can be used with any kind of PC/Laptop with minimum of 500Mb RAM. This is a very lightweight application and can be used for visualisation of almost any places in the world.

3.2.5.2 Safety requirements

This application uses the Foursquare ID and Foursquare Secrets which helps in creating individual account for users to use their own secret and ID. This is a client server model of application. The entire data lies in the server, so even if there is any damage of data, or loss of data, they will have a back up. The GUI is on the client side, which contains nothing except the data. The request is send and the data is got back, cleaned and visualised. If there is any extreme damage to a wide portion of the database in the server side due to catastrophic failure, such as disk crash, the recovery method resumes part of the copy of the database that is backed up in the archival storage generally in the form of tape and hard drive. So Wisp ensures safety for the users.

3.2.5.3 Security Requirements

Since the foursquare API is a genuinely strong API with individual key and token, so it satisfies the security of the individuals very easily. So, we need not make a login system for every user, the token servers as a login system for every user.

3.2.5.4 Software Quality attributes

There are the following quality attributes that the Wisp application provides for all the users:

- Availability – the foursquare API is a good API, maintained by a community of trusted users which verify and cross check each data. So the data is genuine and is available for 24 hrs a day, since there is a rate limit for regular users, a user can make about 10000 requests per day, which is too much for any user.

- Correctness – the foursquare API is used by almost 150,000 developers, so the data needs to be correct else foursquare wouldn't have been so much popular.

- Maintainability – since we have used modular design for the building and creation of Wisp, it is easily upgradable and maintainable. It will be open sourced and will help a lot of developers to collaborate and give it a new life.

- Usability – the Wisp application is user friendly. The UI is self-intuitive. Wisp also comes with a help option which gives the extra arguments that can be passed to wisp in command line for more features.

3.3 Planning and Scheduling

This is the involvement of the time that is required to be given by the developers and other people involved in the project. There are two popular types of charts that are used by the industry. They are pert and Gantt charts.

3.3.1 PERT Charts

Pert chats are used before a project begins to plan **[11]** and determine the duration of each tasks. Gantt charts are used during the development of the project, when a task is broken down into smaller pieces of tasks. Gantt charts highlights scheduling constraints. It stands for Program Evaluation and Review technique. A PERT chart illustrate project as a network diagram. The US navy created this tool in the 1950's as they developed the Polaris missile. Project managers use the PERT chart to analyse the responsibilities and tasks and to determine the minimum time necessary for the completion of the project.

We have used similar techniques to design our first ever PERT chart. We have divided the project into seven main parts. Since our project is not so much advanced, so we have used only seven stages to develop the first prototype.

The steps are:

- Create Software Specification – this is the necessary and the first step required in a Software Development Life Cycle. The main aim of this step is to carefully understand the problem and find the needs that are required to execute the problem. Our application is simple, so we started this at 10th December, 2018. The main objectives of this step were to find all the necessary tools that were necessary to build the Wisp application. We found that the Python 3 programming language was much more suitable than other tools available for the creation of the project. We also took notes of the libraries that would have been required to make such application. It took about 2 days to create the software requirements and specifications.

- Creating the Backend – then comes the creation of the backend for the actual development of the software. The backend is created using the Python 3 language. We have found that there is a good flexibility in creating and merging both the backend and the front end using the same language. We started the creation of the backend at 21.12.2018 soon after the completion of the Software Requirement and Specification document. The creation of the backend was necessary before the creation of the Graphical User Interface (GUI), since we would get the Command Line Interface (CLI) at the very beginning. After that we gained confidence about the feasibility of the project and we estimated the total cost and time required for the project completion. It is necessary to create a backend for rapid prototyping of the initial stage of the software. It took about 5 days in total to create the back end.

- Check and Test back end code – after the creation of backend we checked the backend code for errors. We used modular approach to build and test the code. So, it took about 7 days to complete this part of the lifecycle. We started soon after the completion of the backend, i.e., we started at 27th December, 2018. We have implemented various changes and debug mode in creation of this test phase. This helped us a lot in future development of the project, and ensured that the project can be maintained and developed in the near future.

- Design of the GUI – this could have done in parallel with the creation of the backend code. We first created a skeleton for the GUI using modular approach and after the testing of the GUI which was done along with the creation; we merged the GUI with the backend. This took about 4 days to complete in total.

- Release of First prototype – this was done after all the testing and integration of the modules were completed. This was started at 09.01.2019 and took only 2 days to complete. This is the fun of modular testing, that we can integrate all the modules with ease.

- Documentation and Open Sourcing the Software - this was done after the creation of the first prototype and was released to the open source. The link to the repository is given below **[1]**. The main objective for open sourcing the software was that we get more collaborators from the open source community and help to improve the software and maintain the existing code. We can improve this software in the form of GUI, the clustering algorithm that it uses and other things. We can also create our own module for advanced and AI based clustering rather than general clustering in the near future.

This was the steps in general in the creation of the PERT diagram of the software as shown in **Figure 3**. The advantage of using the PERT diagram was to lay a blueprint in creation of the software. This includes the actual roles and responsibilities of the developers working in the project. The main aim of creating PERT was to estimate the earliest time to complete the project in all possible circumstances. We actually followed the diagram and made the necessary changes to the software.

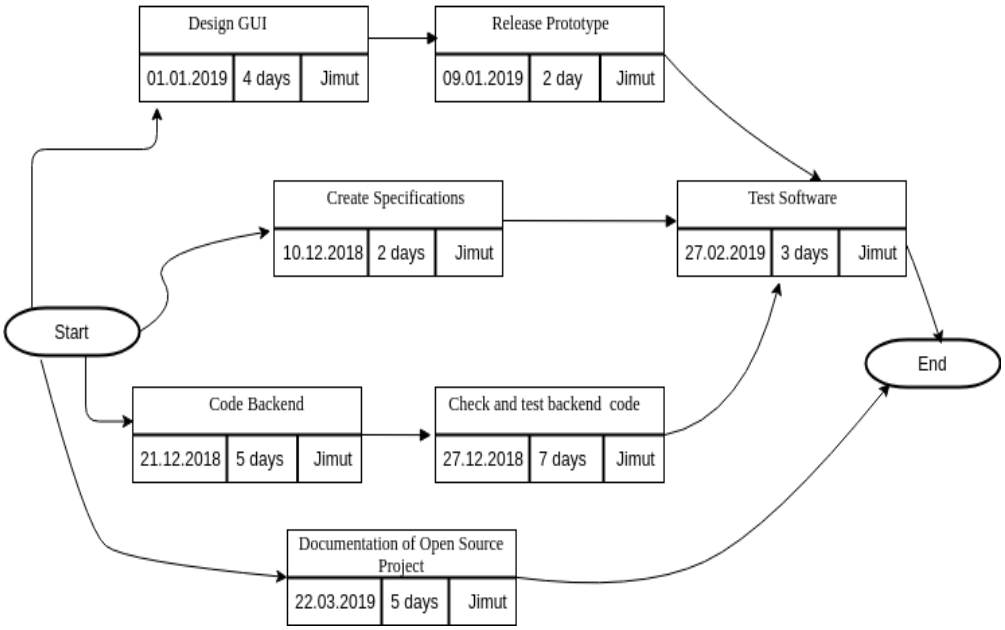

Figure 3: Pert Diagram in the creation of Wisp application

3.3.2 GANTT charts

Gantt charts are a type of bar charts that are used during the development of the project. It is named after its inventor Henry Gantt **[12].** This chart lists the tasks to be performed in the vertical axis and the time intervals on the horizontal axis. The duration of each of the activities are represented by the width of the horizontal bars. It summaries the project by actually plotting the start and end dates of the project.

We have used similar techniques to design our Gantt chart for the project after the completion of the PERT chart **[13]**. The Gantt chart for designing of Wisp is shown in **Figure 4.** The stages are similar to the PERT chart, but they are actually more emphasised on the time required.

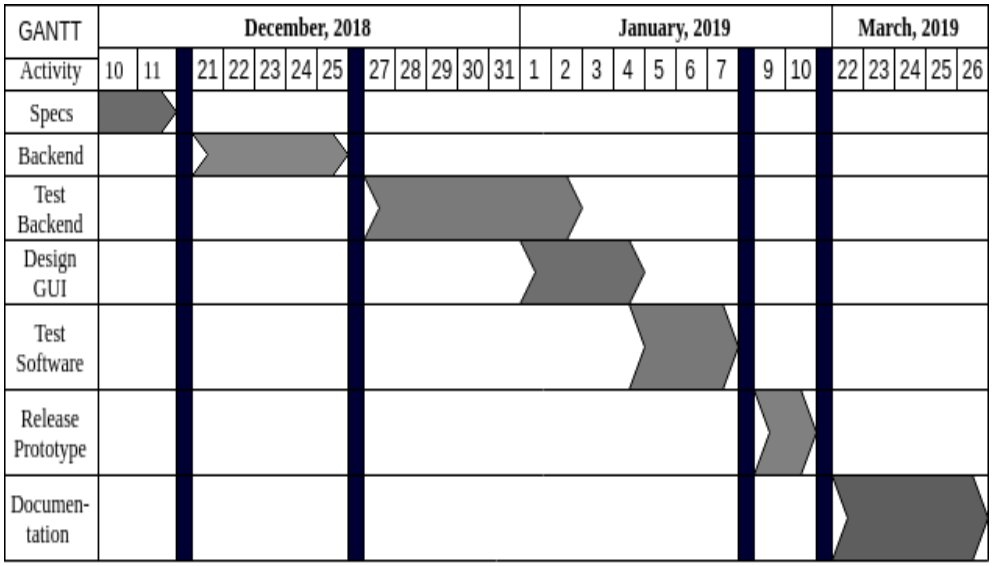

Figure 4: Gantt chart for designing of Wisp Application software

We got a good visual **[14]** of the time and cost that will be required to create such project. The specifications were designed and noted down very fast, i.e. in approximately 2 days. The Backend was coded before the GUI of the application because we needed a working model of the CLI of the application for creating confidence. We took the most of the time and design issues for creation of the GUI of such software. The backend was coded from scratch and was tested using modular testing. We found too many bugs and cleared too many bugs. One of the bugs was the time out bug. The main problem with this bug was unknown. We were using good quality bandwidth using our 4G Jio network. The software was running smoothly and fast during CLI, when we integrated the GUI with the CLI, we found that the software was not working as it should do. It was sending requests and wasn't getting or fetching any raw JSON data. Then after a lot of stack overflow we realised that due to bandwidth minimisation, it was actually time outing and wasn't getting any data from the API. So we added a time out of 10 seconds. We coded the GUI using the famous tkinter library in Python 3. It was done quickly and the first version of the software was completed. It was then integrated and the first prototype was made after that. After the creation of the prototype we created the documentation for the application. The software is open-sourced and is available on Github **[1]**, under my account.

3.4 Software and Hardware Requirements

The testing, development of the Wisp application is done in Linux.

The software requirements are:

- requests==2.21.0
- pandas==0.24.1
- folium==0.7.0
- geopy==1.18.1
- numpy==1.16.1
- wget==3.2
- datetime (latest inbuilt version)
- IPython (latest inbuilt version)
- pip (latest inbuilt version)

We have provided a file called as requirements.txt which has all the software dependencies required in it. All the dependencies can be installed at one-go using **pip install –r requirements.txt.** This function is inbuilt in Python and any kind of requirements can be installed recursively from a text file.

The hardware requirements are:

Architecture: x86_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

Address sizes: 39 bits physical, 48 bits virtual

CPU(s): 4

On-line CPU(s) list: 0-3

Thread(s) per core: 2

Core(s) per socket: 2

Socket(s): 1

NUMA node(s): 1

Vendor ID: GenuineIntel

CPU family: 6

Model: 78

Model name: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz

Stepping: 3

CPU MHz: 500.002

CPU max MHz: 2800.0000

CPU min MHz: 400.0000

BogoMIPS: 4801.00

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 3072K

NUMA node0 CPU(s): 0-3

Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp flush_l1d

This requirement is generated by pressing the command lscpu in the terminal. The minimum requirement required for running of Wisp application is 500 Mb of RAM with 100 MB hard disk. The Wisp application is a very light weight application since it uses the lightweight python framework known as tkinter for the GUI.

3.5 Preliminary Product Description

What we have designed is a standalone application based on python programming language. It eliminates the problem for the user to do massive research works according to their preferences and helps them to find the best places according their needs in real time. It clusters the preferences of the users in a folium map. It also takes the data from an API known as Foursquare API

[2], which has active community to update the data very frequently and also it is trustworthy API. There are two versions of the API, one is free, which is fine for one account of user for domestic uses and another is premium version for unlimited requests, generally designed for Organizational uses. The server for the API remains active for 24 hours a day, so it is very trustworthy and helpful for the user to use it any time. We build a full-fledged application for solving this problem of manually researching the places and we provided some extra features in our application. These extra features will help the users to use the application more effectively.

Some of the extra features are: -

1. There is wide variety of maps to select from.

2. We can visualise the output map in real time and also save it for later use in HTML file.

3. We have provided a secrets.txt file so that we don't have to manually insert the foursquare secrets and ID each time.

4. There are unlimited preferences that can be entered.

5. We can enter the location of almost any places and we can focus on those places only which forms a radius within that location. Both are inserted manually in the application.

6. We take the data from community maintained and rich API, called the foursquare API. We think that if we promote such API then there will be a lot of improvement in the actual data content of the API, which in turn will help in the data and performance of the application. More users will make an account and more amounts of data would be feed in terms of reviews comments etc. into the API.

3.6 Conceptual Models

We have used agile methodologies **[15]** in providing alteration to the existing project. Agile is generally used in business to predict the outcome in advanced. It reduces the development cost significantly; since significant

amount of the time is spend in planning and the later in execution. This helps in delivering the system quickly.

3.6.1 RAD model

We have used Rapid action development (RAD) model for developing Wisp. The generic diagram of the RAD model is shown in **Figure 5.** It is a type of incremental model **[16]**; each module is done in parallel. It is done in such a way as if each of the modules was a mini project.

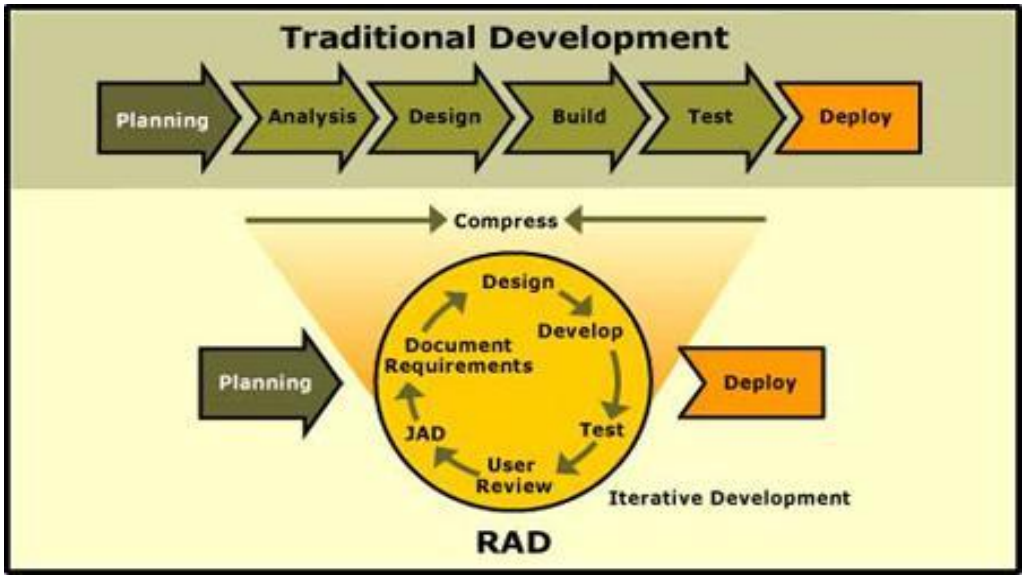

Figure 5: Rapid Action Development (RAD) model.

There are time limits in which a certain module should be developed, delivered and then assembled into a working prototype. This helps the developers and the clients to get a working prototype at each stage. This also helps the developers to get feedback on the bugs and issues that arise during the completion of intermediate prototypes, and accordingly modify and repair those bugs.

The advantages of RAD model:

1. RAD model reduces the total development time.

2. Increases reusability of components, since each prototype is developed after the end of one cycle of iteration.

3. Quick reviews occur at the end of each prototype.

4. It encourages client feedback and helps the developers to work on the bugs.

5. Integration from very beginning solves a lot of integration issues.

The advantage of using RAD architecture in System design is it effectively handles unclear requirements, unfamiliar technology and complex systems in an efficient way. This method is reliable when all requirements are initially known. Schedule visibility of RAD model is excellent, which helps to divide tasks among developers. When a requirement is unclear, one can easily throw away the last made prototype which is rejected by the client, or was created and was unsatisfactory. It is best suited for short time schedules, since RAD increases the speed of development. It moves away many of the critical design decisions earlier in the project; consequently, this helps the developers and the project managers to address the risk involved early and improve on the risks by providing strategic solutions. In certain cases, the requirements are clear; the company can also automate the tasks using certain tools, like automate the generation of web pages by adding dynamic content to it. The RAD model basically helps to manage the system, and come up with an effective plan to build and deploy the project in minimum time and cost.

Different phases **[17]** of RAD model:

- Business model – this is designed on the basis of the flow of information and distribution between various business channels.

- Data modelling – The information that is gathered during business modelling is refined to a set of data objects during this phase, that will help in business in the near future.

- Process modelling – in this phase, the actual information flow occurs from the model that is implemented and build during the previous phase.

- Application Generation – tools and automated software are used to convert the model to a prototype.

- Testing and turnover – since there is individual testing of prototype for each iteration of the model, so the overall testing time is reduced significantly in a RAD model.

CHAPTER 4: SYSTEM DESIGN

4.1 Basic Modules

There were several of the modules required during the building of the Wisp application. Those modules were not inbuilt in Python 3 but could be installed from PIP, the Python package manager, which is OS independent.

Some of the main modules that were necessary during the building of the Wisp application are given below:

4.1.1 Requests==2.21.0

Requests allow any user to send *organic* or *grass-fed* HTTP/1.1 requests. It is usually done for web scraping. This can send request to a web server and we can get back the content easily and without the need for manual labour. This technology is better than existing technologies because this requires minimum amount of code to perform the operation. There's no need to manually add query strings to the URLs. If it was C++, then for performing requests to a web server we needed to write documents after documents of code to do a simple job. We needed to form-encode our POST data. Keep-alive and HTTP connection pooling are fully automatic **[18]**.

Besides, all the developers using it, it is one of the most downloaded Python packages of all time, pulling in over 11,000,000 download every month from different systems. This is one of the simplest libraries in Python yet performs a variety of jobs from web scraping to retrieval of data from any remote web server. A lot of people use this library for doing their development works.

User Testimonial **[19]** says that Spotify, Microsoft, Reddit, The NSA, Her Majesty's Government, Google, PayPal, NPR, Kippt, Sony, Amazon, Nike, Twilio, Runscope, Mozilla, Twitter, Lyft, BuzzFeed, Heroku, Obama for

America, Transifex, Native Instruments, The Washington Post, SoundCloud, and Federal U.S. Institutions that prefer to be unnamed claim for using Requests internally. The official logo of Requests library is shown in **Figure 6.**

Figure 6: The official logo of Requests library in Python **[19]**.

4.1.2 Pandas==0.24.1

Pandas is a Python module which provides fast, expressive and flexible data structures. It is designed to work with structured which are tabular, potentially heterogeneous, and multidimensional data structures; and also time series data both easily and intuitively. It aims itself to be the most fundamental high-level building block for doing real world raw data analysis and data manipulation in Python. It also has the broader goal of becoming the most powerful and flexible open source data manipulation tool available in any language. It is already well on its way toward this goal **[20]**. It is used by Analytics and developers alike. We have used this because the Folium library provides easy data fetching with this library.

There are two primary data structures in the Pandas library. They are Series which is 1 dimensional and DataFrame which is 2 dimensional. These two data structures handle the problems of almost every field of engineering's,

ranging from Social Sciences, finance, statistics and other fields where the use of CSV is must. The DataFrame is superior to the data.frame library in R. It is built in top of Numpy library and is intended to integrate over many other third party libraries.

The additional features of the Pandas library are:

1. This handles missing data with the NaN, both in floating point and non-floating point.

2. The size of the data frame and higher dimensional data structures can be mutated easily by adding and removing columns.

3. The data can be aligned automatically with the set of labels.

4. There is almost all the SQL functions present in Pandas, which are optimised and very fast.

5. It does labelling, slicing and subsets large dataset easily.

6. It is used to reshape, handle a lot of varied data structures present in CSV files etc., in a rapid manner. In other words Pandas is the most sought library out there which performs all the data manipulation and other tasks very rapidly. It is used by the data scientist and analysts alike.

Additionally pandas is sponsored by NumFocus, and released under BSD-licensed library. It is an open source library and is improving rapidly over the years. It has gained popularity over the years and is used by a lot of researchers. Due to the open sourcing of the library, this has gained a tremendous popularity in the past few years.

Out[5]:

	name	categories	address	cc	city	country	distance	formattedAddress	labeledLatLngs	lat	lng	postalCode	state
0	Naturals Ice Cream	Ice Cream Shop	77/1A Park Street	IN	Kolkata	India	2077	[77/1A Park Street, Kolkata 700016, West Benga...	[{'label': 'display', 'lat': 22.550341, 'lng':...	22.550341	88.354897	700016	West Bengal
1	Metro Ice Cream	Ice Cream Shop	NaN	IN	NaN	India	1646	[India]	[{'label': 'display', 'lat': 22.553568, 'lng':...	22.553568	88.352151	NaN	NaN
2	Thanco's Natural Ice Cream	Ice Cream Shop	18A, Ram Mohan Dutta Rd	IN	Kolkata	India	3742	[18A, Ram Mohan Dutta Rd, Kolkata 700020, West...	[{'label': 'display', 'lat': 22.53427080110658...	22.534271	88.351021	700020	West Bengal
3	The Cream & Fudge Factory	Ice Cream Shop	3-A, Humayun Pl	IN	Kolkata	India	907	[3-A, Humayun Pl, Kolkata 700087, West Bengal,...	[{'label': 'display', 'lat': 22.56065779513747...	22.560658	88.351954	700087	West Bengal
4	Fire N Ice	Italian Restaurant	41, Jawaharlal Nehru Rd	IN	Kolkata	India	2050	[41, Jawaharlal Nehru Rd, Kolkata 700071, West...	[{'label': 'display', 'lat': 22.54940945481822...	22.549409	88.349471	700071	West Bengal

Figure 7: A dataframe in Jupyter notebook using Pandas

4.1.3 Folium==0.7.0

The folium library is built on the data wrangling superiority of the Python language and the visualisation **[21]** and the mapping strength of the Leaflet.js library. We can manipulate our data using Python's library and then visualise with easy using the folium library. The folium library uses Maps from the leaflet.js library, which is the optimised version of maps in the JS coding language. The data can be plotted on the interactive leaflet map. It provides both binding of data to a Map as well as provides custom made marker which will pop up the visualisations during click to make it more interactive to the users.

The Library has a number of tilesets, which are inbuilt in folium. It ranges from OpenStreetMap, MapQuest Open, MapQuest Open Aerial, Mapbox, and Stamen. It also supports custom made tilesets with Mapbox or Cloudmade API keys. It supports both GeoJson and TopoJson overlays. The visualisations are used to do statistical study and other developing activities. We have used it because the data from the API could be easily converted to the DataFrame and then used using the folium map. The flexibility of the map with the data was superior for our visualisations.

Figure 8: The official folium logo **[21]**

4.1.4 Geopy==1.18.1

Geopy is a client in Python 2 and Python 3 for several popular geocoding services. It makes easy for developers to get the latitude and longitude of addresses, cities, countries and landmarks easily using third party geocoders and additional data sources across the Globe. This includes the geocoder classes for OpenStreetMap, Nominatim and Google API along with other geocoding services. It can also help to calculate the distance between two points in a Map, using set of inbuilt functions.

Figure 9: The official Logo of Geopy Library in Python **[22]**

4.1.5 Numpy==1.16.1

Numpy is the library for Python programming language which provides a huge support for large, multi-dimensional arrays **[23]** along with matrices. It has high level mathematical functions to operate in these arrays. It was originally created by Jim Hugunin with contributions from several other developers. It is an open source library and has many contributions. Python programming was not for mathematical computing, but after the creation of the numpy library it attracted a lot of developers and scientists for doing mathematical operations using this high level library in python.

Numpy refers the CPython implementation of Python. CPython is a non-optimising byte code interpreter. The algorithms that are run with Cpython are much slower than its compiled equivalent. Numpy gives functionality comparable to Matlab, since they are interpreted alike. They both allow users to run programs way faster than other languages as long as the users are using arrays or matrices. When scalars are used, Numpy becomes comparatively much slower. Numpy relies on lower sub-routines like BLAS and LAPACK which are used for faster computation in Linear Algebra.

Since Numpy uses arrays, indexing, slicing is much more easily using inbuilt Python features, the image recognition and analysis becomes easier. Numpy has been optimised a lot in the past few decades and is developing more and more. It has become the attraction for using python to solve mathematical problems more efficiently.

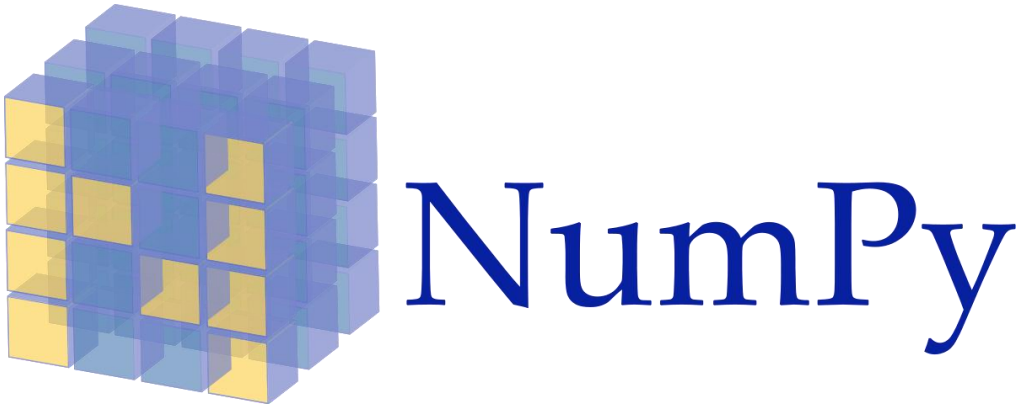

Figure 10: The official logo of the Numpy **[23]**

4.1.6 IPython

IPython is a command shell. It is used for interactive computing in multiple programming languages. It provides tab completion, introspection, rich media and shell syntax. It was built because of its ability to examine to examine the type and properties of an object in runtime. It also provides interactive visualisation on the shell. It is based on the architecture that provides parallel and distributive computing.

The basic testing and visualisation of the Wisp was done in IPython. Then it was integrated with the actual code later. The modular testing was done in IPython. IPython provides a lot of interactivity during the run time of code. The main feature of IPython is its generic ability to visualise objects and data.

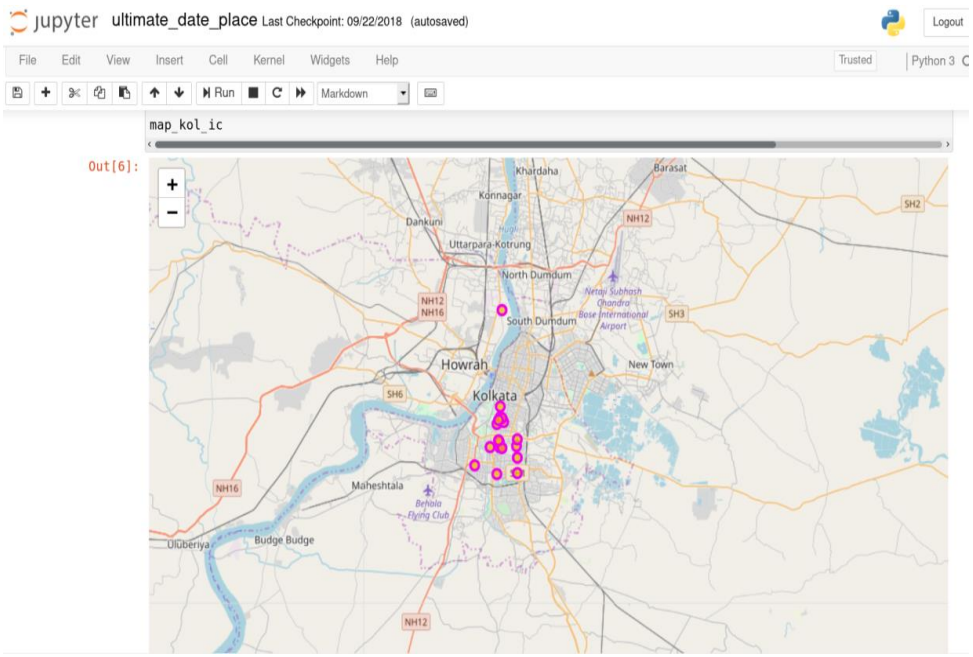

Figure 11: Figure of Ipython Interactive Shell in Jupyter Notebook

4.2 Data Design

The database design forms an essential and crucial step in the development of any software **[24]**. It is the organisation of data according to any database model. The designer determines the flow of data between modules, how the data are stored, how they interrelate. The theoretical representation of data is known as ontology. The person who is actually doing database design shall have to be a master to design it correctly, utilising less resource, fetching the data fast from the database, etc. comes into play. This design of database is done with the help of the Software Requirement Specification (SRS) document.

Once the designer knows what data to be used, he then determines the relationships between the data in the database. He determines what data is needed, how the dependencies between data exists, etc. He is needed to know how one data changes or affects another data in the database. How the data should affect, and how some data should be prevented from affecting during a certain time. After that, he determines how logically the data is stored. How he could minimize data redundancy without the performance of the whole system.

The steps which a designer should follow to design database are:

- Should determine the purpose of the database at first, which in turn helps the preceding steps in the design of the database.
- Find and organise the information required into blueprints which shall help to map the schema of the database. He should find all the data relation from the models. He should know how the data depends among them.
- Divide the information into major entities known as tables.
- Turn information into entries in the table. Turn data into column.
- Choose the primary key wisely by seeing the data which shall uniquely determine each row of the table. This is done by looking all the entries or maybe done by combining one or more entries in the table so that they form a unique ID when combined together.
- Should define the relationships between all the tables. How the foreign key should relate to a primary key in a table and all those tedious jobs are needed to be done after looking at the database.
- Apply normalisation, i.e., apply minimisation of the redundant data. There are different forms of normalisation that could be applied to a database. It depends on the expertise of the database designer how he develops the database so that the data doesn't hamper the speed and efficiency of fetching the data from the database.

Since Wisp doesn't use a database of its own as such, it uses the database present in the Foursquare API. The database of the foursquare API is as follows:

4.2.1 Foursquare Database Design

This schema has been found in Github **[25],** and is hopefully the schema design for the foursquare API interior.

Table: **Places**

A table that houses all of the places a user can check into.

place_id: (Integer), Primary ID that preferably auto increments (if supported in chosen DB).

name: (String), Name of the place.

address: (String), Address of the place.

city: (String), City of the place.

zip: (String), Not all zip/postal codes are numeric.

tags: (String), Preferably a JSON encoded string of tags.

latitude: (Decimal 9,6), The latitude of the place.

longitude: (Decimal 9,6), The longitude of the place.

Table: Checkins

checkin_id: (Integer), Primary ID that preferably auto increments (if supported in chosen DB)

user_id: (Integer), ID of the user who checked in

place_id: (Integer), ID of the place the user is checking into

datetime_added: (Datetime or Timestamp Integer), When did this checkin take place?

Table: Tips

A table housing all tips left for a particular place by a user

tip_id: (Integer), Primary ID that preferably auto increments (if supported in chosen DB)

user_id: (Integer), ID of the user who left this tip

datetime_added: (Datetime or Timestamp Integer), When was this tip added?

tip: (String), Text description of the tip

4.3 Schema Design and Data Integrity Constraints

After the database design, comes the schema design. The Wisp application doesn't have a schema of its own. It fetches data from the foursquare API. The probable schema for the FourSquare database is shown in **Figure 12.**

The database schema **[27]** of an application system is the structure of the database described in formal language. It forms a blueprint to see how the database is formatted. It shows the tables, the relationship between different tables etc. The database schema plays almost same role as the predicate calculus. The designer who is designing the database schema should have enormous knowledge in the field of database design to come up with a good design for a large organisation.

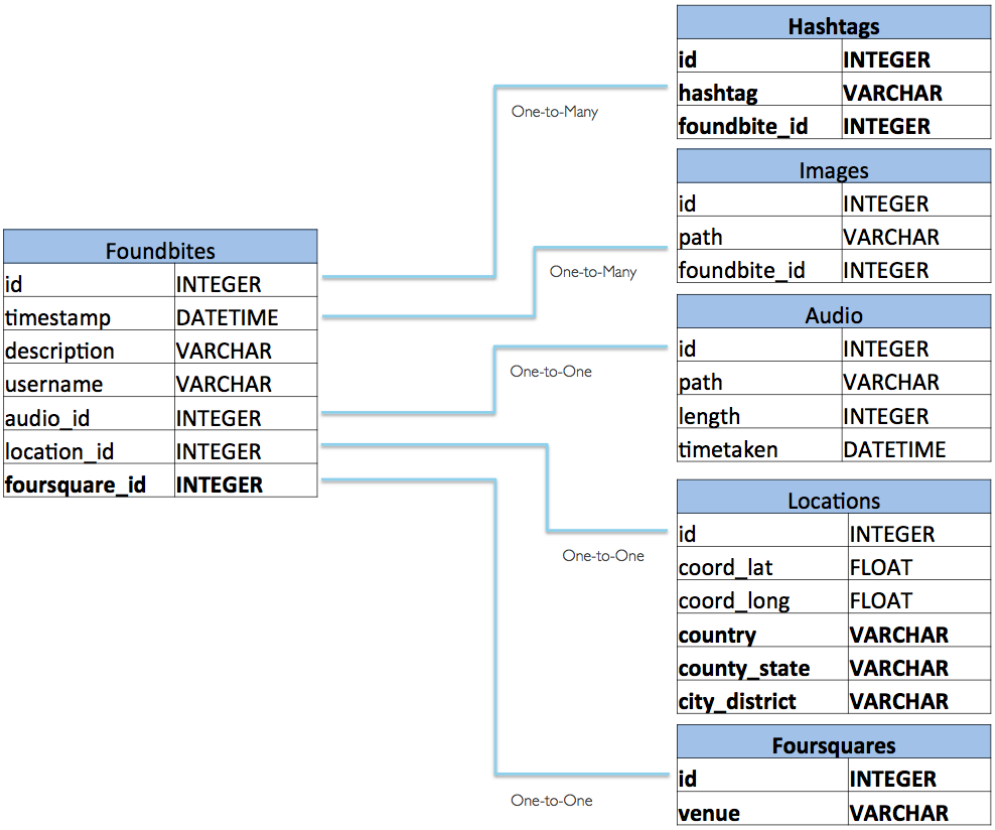

Figure 12: The Schema for FourSquare API **[26]**

We see from the above figure that the schema of the database is designed in such a way that it forms one-to-one relationship with the tables. The database schema design is done in its minimalism, to prevent the use of redundant data in the database.

Data integrity is the assurance **[28]** of the data over its entire life cycle. It is ensured that the data is retrieved in the same order as it was given in the database. The main goal here is to minimise the storage of the data in the server. Any changes to the data due to certain unavoidable circumstances such as hijacking of data, hard disk failure, human error, etc., results in the failure of the data integrity. There are two types of integrity, Physical and Logical. The former focuses on the fact how to store the data without loss. It generally includes the checking of hard disks etc. so that due to no physical constraints does the data gets lost in the process. The later refers to the process in which the data can be stored and minimised using logics and predicate calculus. Since the diagram in **Figure 12,** is done by experts in the industry, so we can trust that it follows data integrity principles internally inside the foursquare API.

4.4 Procedural Designs, Logic Diagram and Data Structures

The control flow diagram of the wisp application is shown in **Figure 13.**

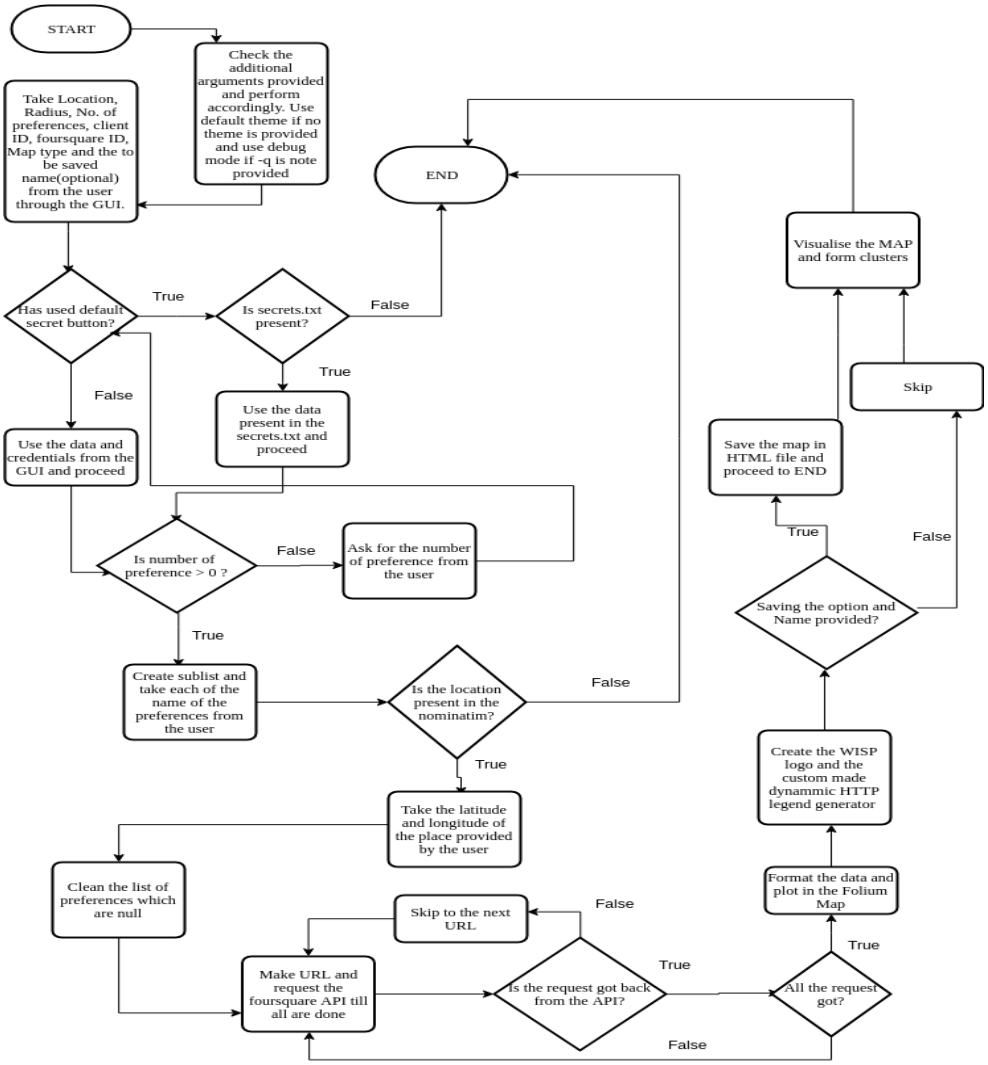

Figure 13: The Control Flow Diagram of the wisp application

The control flow diagram helps us to understand the flow and details of the process that is going on. It helps to visualise the breaks that happens in a software designing paradigm, where the control deviates and takes new turn, where the control ends etc. This helps us to see the inputs and the outputs in a data flow diagram. It makes the roadmap of the application. The control flow and the data flow diagram of wisp are a little bit complicated.

Many steps have been skipped to save the space for the creation of the diagram. When the application starts, it checks whether the additional arguments have been provided to it or not. It checks the theme that has been provided to it or not. It uses the default theme if no theme is present. It also checks if the debug mode is enabled or not. Debug mode creates the retro Banner of Wisp and the details of the data that are fetched along with the app running at each instant. This mode has helped in the creation of the Wisp Application software. The checks are done. Additionally, it checks whether the

help or version arguments are given or not, if it is given then it performs the tasks according to the arguments and does nothing extra. When the wisp application starts with the theme present in it, it takes the data from the user. There are three buttons that will be present. One is the use default secret button. The use default secret button will help to use the secrets i.e., foursquare ID and foursquare secrets from the text file secrets.txt present. The Software quits if it doesn't find such file in the directory from where the wisp application is executed and the default secret button is used. If the file is present, then the data is read from the JSON reader. The number of preferences, location, and radius from the location are taken. After that, the sub list is formed which takes the name of the preferences present. The map type is taken and it is by default MapBox Bright. The name of the optional saving argument is also taken in case the rendered map is to be saved for future use. The location is checked using the geocoder API and if it is not found then the application is ended. If the location is found, the latitude and longitude is fetched for making the requests.

The lists of preferences are cleaned, in case if the user forgets to put some of the preferences or the required numbers of preferences are not given. The application then makes URL according to the location, preferences and radius, and makes request to the foursquare API until the entire URL are used. The data is the converted to pandas dataframe for cleaning and then to Numpy array. It is then plotted in the folium Map and saved, if the name of the optional argument was provided before in the GUI. The Custom made legend generator is made, and the additional things are carried out and it is visualised using a custom made HTTP server. After the visualisation is over the application comes to an end on quitting it.

4.5 User Interface Design

The user interface is designed using tkinter **[29]** library in Python 3. It is considered as a de-facto standard GUI library for Python 3. It is popular because of its object oriented paradigms and the user-friendliness to design almost any type of GUI using it. There are different parts of a tkinter based GUI application such as frame, label, textbox, canvas, scrollbar, buttons etc. The alignment can be done with the help of grid and pack in tkinter. The applications produced in tkinter are easy to make and are customisable.

We have provided three themes for the WISP application. The first one being the default theme, also known as Jimut's Classic theme as shown in **Figure 14.**

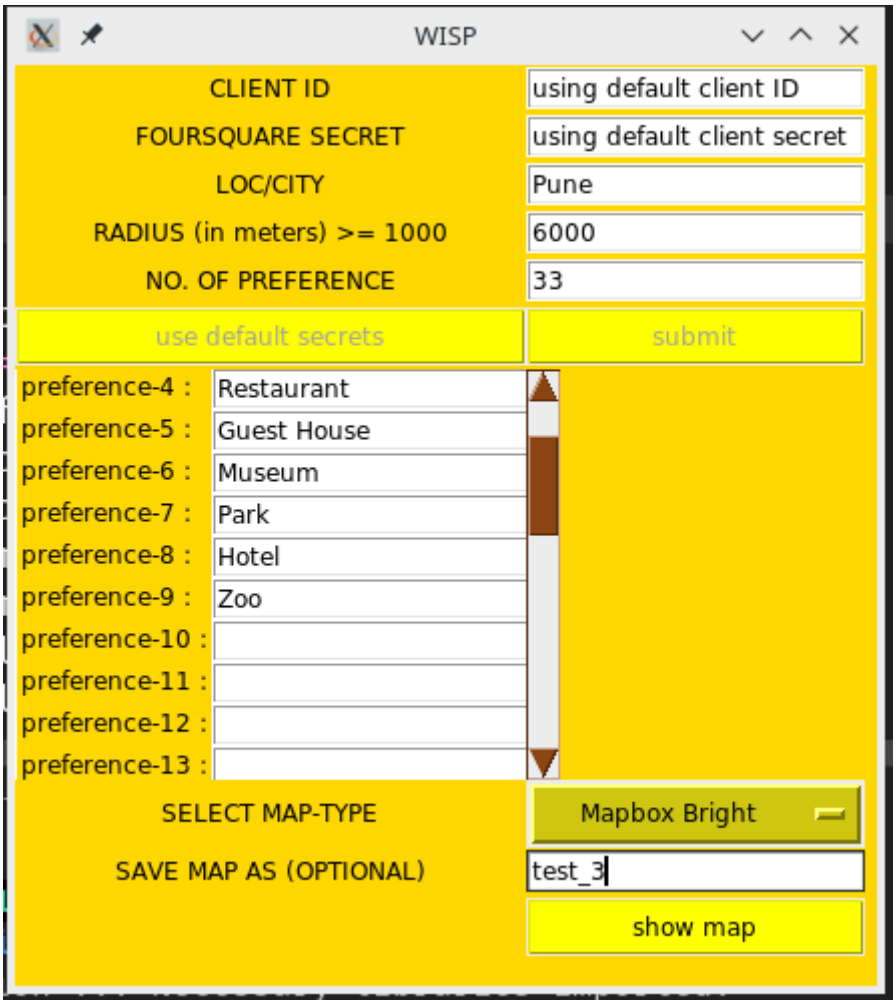

Figure 14: Jimut's classic default theme for wisp given by additional argument as –t 1 or none

The classic theme is by default present and set for the wisp application. Another theme is Jimut's light theme as shown in **Figure 15.**

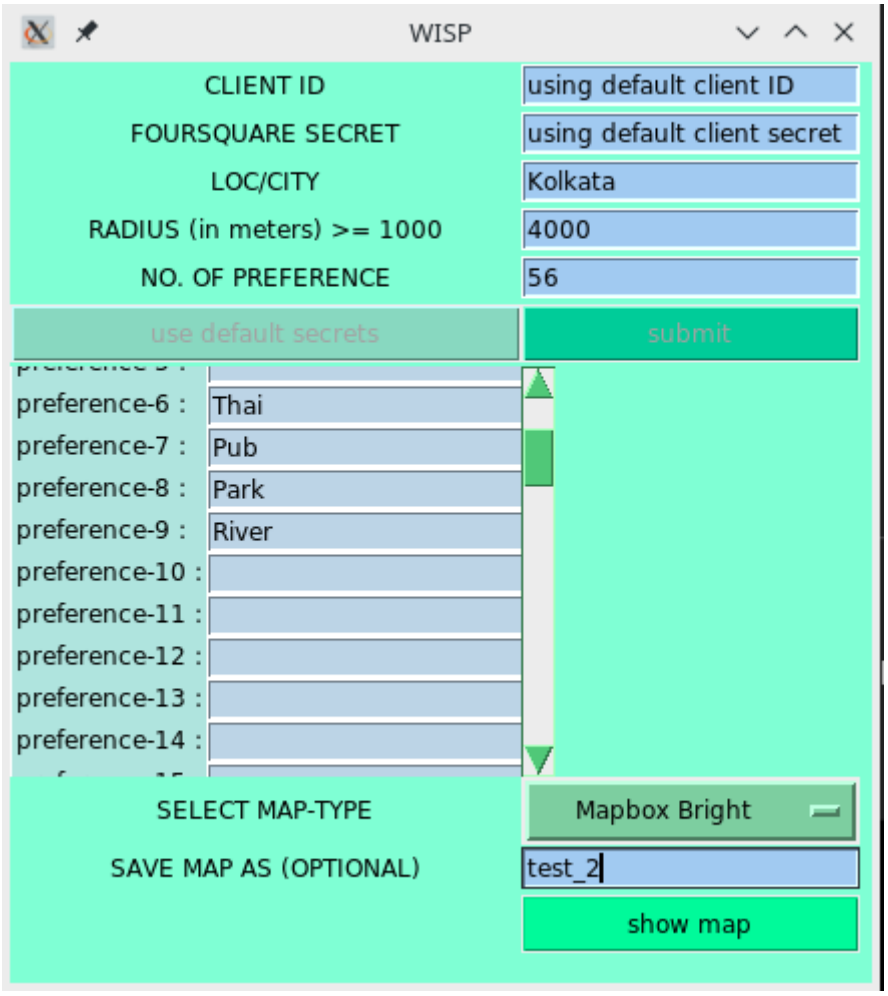

Figure 15: Jimut's light theme for Wisp, can be obtained by passing argument -t 2 in the terminal

This light theme can be obtained by passing the argument –t 2 along with the wisp command in the terminal. These are done to make the application more appealing to the user. We can set the background and foreground colours by passing a parameter as an additional argument. These themes can also be changed by manipulating the source code. The code is self-intuitive and can be changed by the user. The colours are given in hexadecimal format for example "#FFFFFF" will represent white colour and "#000000" will represent black colour. This is a traditional way of changing colours in HTML and other applications where User experience forms a key role in the development of the application.

Another type of theme that is provided in the Wisp application and can be changed by passing an argument of –t 3 along with the wisp command in the terminal is the Jimut's dark theme as shown in **Figure 16.**

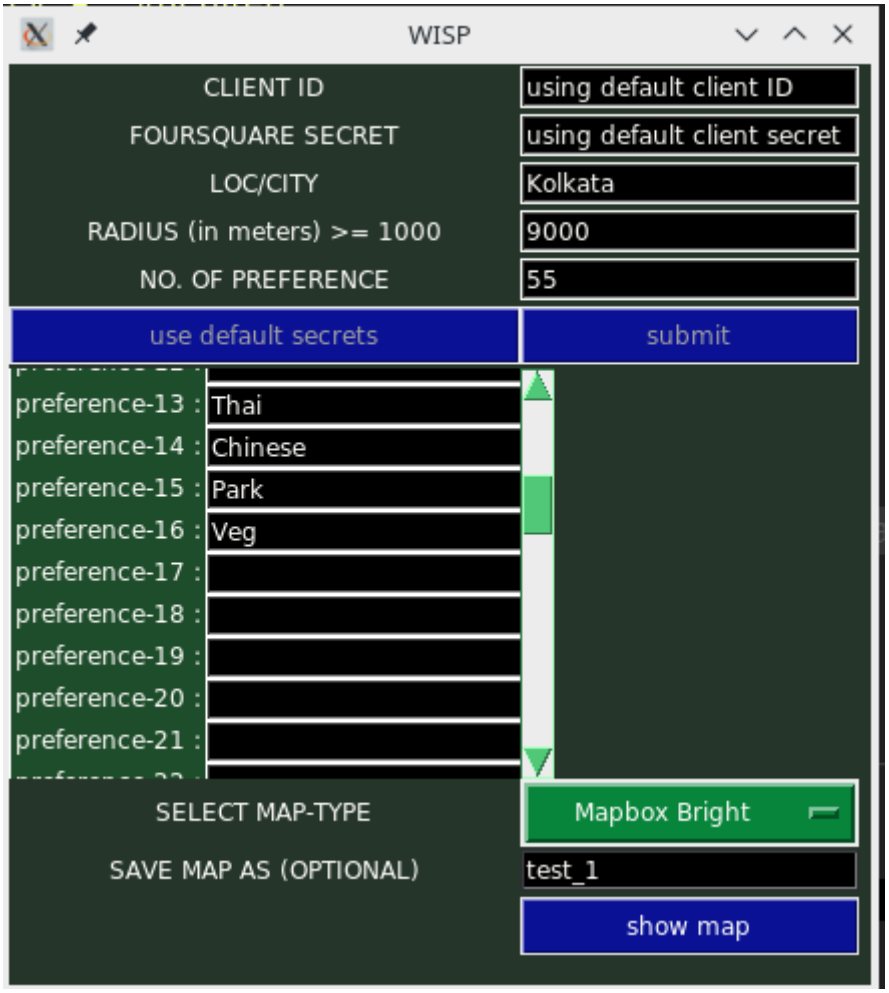

Figure 16: Jimut's dark theme for Wisp, can be obtained by passing argument -t 3 in the terminal

Wisp is a fully fledged Linux application, which is O.S. independent. The application when viewed from terminal looks something like this, as shown in **Figure 17.** This is the spirit of programming in Linux, and the terminal can make the true Linux lover fall in love with the wisp like any of the open source applications. There are four arguments that can be provided in the wisp application. The first one is –h or --help for showing the help message as shown in **Figure 17.** The next one being –q or --quiet, which will bypass the debug mode and provide a clean terminal for the user. The next one being –t or --theme mode, which changes the theme. There are three themes present in wisp application. –t 1 will give the Classic theme. –t 2 will give the light theme, and –t 3 will give the dark theme.

```
➜  wisp git:(master) wisp -h
usage: wisp [-h] [-q] [-t {1,2,3}] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -q, --quiet           will not display anything in the terminal
  -t {1,2,3}, --theme {1,2,3}
                        to select theme [1] classic [2] light [3] dark
  -v, -V, --version     displays version
➜  wisp git:(master)
```

Figure 17: The help message of wisp application

The last one being –v or --version mode, which only shows the version of the wisp application and does nothing. The visual element of the software that the user should interact with is known as the UI. It is generally not too much complex to use, and adapted by targeted users. If the user is not able to find the targeted element that they are looking for, then it might be a pain for the users to use that software or application with ease. The user interface should be optimised so that the user can operate quickly in their application according to their needs.

4.6 Security Issues and Test Case Designs

The wisp application doesn't use any kind of database for its own. This is the reason that there are no security issues with this application. There are no chances of hijacking the user's personal data with the help of SQL injection or something alike. The user makes a personal account in the Foursquare API website and gets the key and other credentials from there. There is no question of whether this is a secured method, since the encryption is way beyond to decrypt for us. The only security issue that may come is there might be stealing of the foursquare API key and token, so that should be kept carefully and preserved for future use. The key and token are like passwords for an account, in this case it just serves the purpose of fetching the data of the API from the individual accounts.

We have worked a lot in designing of test cases. We have almost used a spiral model and a RAD system alike to develop this software. The methods that we use are taken from these two testing models. There is a lot to talk about testing and implementation of the Wisp application software; we have discussed that in the next section in brief.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

We have used the spiral model to design and develop the versions of Wisp application. To reduce the potential chaos of developing a software system, we use software paradigms and process **[30]** that help to build high quality software and applications. It describes the tasks that are required for building high quality software applications. Generally, the contents of the paradigms and models that are to be developed are dependent on the nature of the target system. The main benefits that are received by using this approach of software paradigms are, this helps to create the software systematically and using a road map. The methodologies and the blueprints are followed, which helps in calculation and estimation of the total time required for the project to complete.

It may also happen that there was a blueprint that was previously developed by the experienced developers, and one can transfer those skills in the existing software system. This will help in estimation of the risk and cost that will be involved in development of the project. The software engineering paradigm also referred as the software process model or the Software Development Life Cycle is the development strategy that encompasses the process, methods and tools to create software from scratch. The SDLC starts right at the beginning from the software being made, till the end of the software which means, the software being discharged due to the fact that new software has replaced the old software. Hence the software cannot be modified, since it is replaced by current technology. The software development is a structured process. This means that the software development consists of transition steps involved during the up gradation of the software from the current stage to the next stage. The software engineering paradigm and concepts provides a roadmap as a guide to all the engineers who are developing software.

A software paradigm specifies an approach for designing, maintaining and building software. It is a philosophy to design software. Each of the paradigms has their own advantages and disadvantages. This makes a given paradigm suitable and best for a certain kind of software, whereas the other may be suitable for another kind of software. The techniques, methodologies, tools and procedures are dependent heavily on the selected paradigm. There is common process, phases, tasks and activities that are modelled by a software model. These are generally architectural engineering, requirements engineering and requirements analysis and modelling.

5.1.2 Spiral Model

The spiral model was developed by Barry Boehm in 1989, which includes the same steps i.e., requirement analysis, planning, design, coding, testing and delivery as a waterfall model. The motivation behind the creation of the spiral model was that the requirements at the beginning of the project were always unclear, since, requirements and specifications generally evolve over time. The steps always go through a cyclic motion and the requirements and design evolves over time as shown in **Fig 18.**

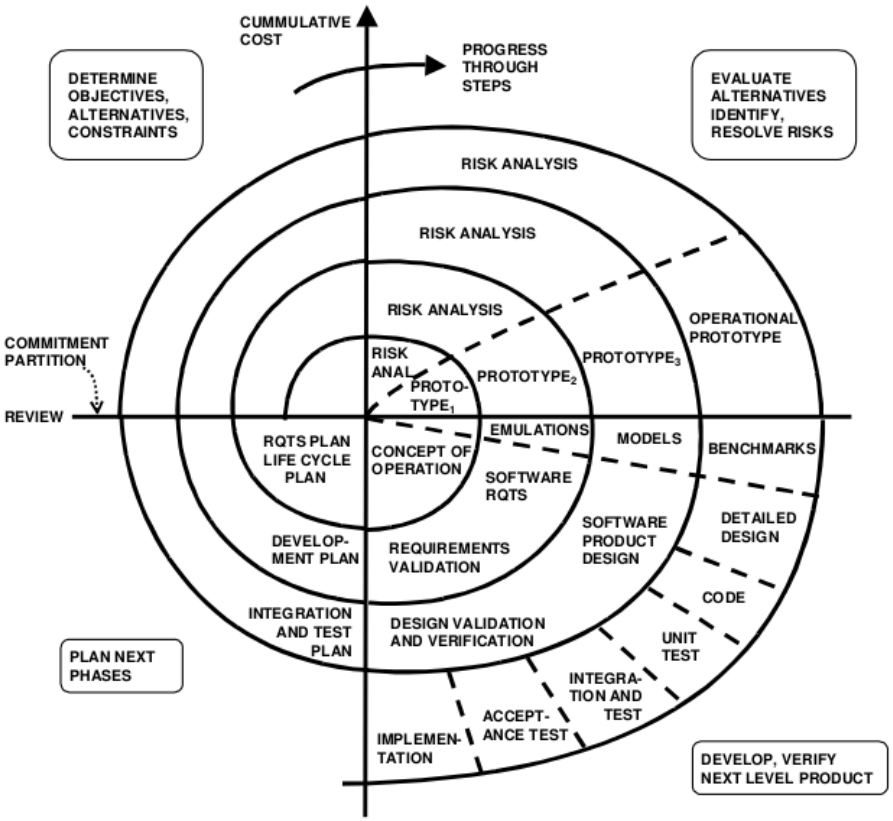

Figure 18: Spiral model in SDLC. Picture Courtesy: XB Marketing **[31].**

The figure shows that the spiral model is non-linear and cyclic. Each cycle of the spiral consists of six phases as in the waterfall model. The angular factor of the spiral shows the progress in the process and the radius shows the cost that is needed to execute the process. It can be described as being sequential and iterative process. Each phase starts with a design goal, after each loop of the cycle, also known as iteration the client receives a prototype of the working model of the software. The main aim of the spiral model is to re-evaluate the

cost and the risk involved in designing of the software. Since this model calculates much of the risk involved in future, so this kind of model is especially suited for projects which have high cost, high risk and the requirements are not clear at the beginning of the project. This kind of model is suitable for large and expensive projects.

There are a lot of advantages of spiral model:

- It takes the change in the requirements of the development process into consideration, which helps to estimate the risk and cost of every prototype that is being made.

- The model is feasible and flexible. It can be used for creating software for variety of situations. Those includes Reuse, component based development and prototyping.

- It can be used as any other model.

- It merges and couples the best practices that are done in the development of waterfall model and rapid prototyping of the software.

- This is for large projects, so it is complicated and a lot of time is used for planning. It is unsuitable for small projects where the requirements and specifications are known at the beginning of the project.

In this phase, the modules, components, architectures all were coupled to form a flexible system. The system was firstly analysed and the modules were made to perform specific tasks. The modularisation of software helps us to test the modules individually and not as a whole. This helps us to debug and test the application in a divide and conquer method. Since testing small codes is much more feasible and easy than to debug documents of code at one time, so this is a great way to design and implement software.

There are four elements of a system that needs to be coupled closely, and are always made in software. They are architecture or blueprint of software. We can use flowchart for easy visualisation of the architecture. Modules handles one specific part of a system, and are present in all the software that we can generally see around us. Components are formed by grouping some modules together and they achieve a single function. Interfaces are the boundary in which the modules and the GUI interact with each other, like a bridge connecting between two parts. Data is the management and flow of data. This is the most important part of software since it provides input and output. Any software is made for manipulation of data. This can be done using modules which takes a form of data and changes to another form. Modules generally take data and

transforms into another form by applying certain transformations according to the algorithm present inside them. Data is the oil or blood of any software system.

5.2 Coding Details and Code Efficiency

We have used the Python 3 programming language to code the Wisp application software from scratch. The code for the wisp.py is present in Github **[1].** If one needs to run the code one could follow the link in the reference. This code is for the wisp.py file. The source code for the main application has been provided below:

```
"""
#   Copyright (c) 2019, Jimut Bahan Pal. All Rights Reserved.
#
#   Please refer to the GNU GENERAL PUBLIC LICENSE for more.
#
#   This is the application (probably basic) to find the location (almost any) in any Country
#   according to the choices of your preference. Uses Foursquare API to get the data (geojson),
#   also uses tkinter GUI for accepting data. Please provide the Access key for the API
#   if bychance not given! This then creates a custom-made http server to visualise the locations
#   in a web browser, because Folium (leaflet.js) doesn't work in GUI or Terminal.
#
#   Caution: Please don't blame me if this doesn't works, cause the data may not be present for
#            some location, since everyone will use free services of foursquare API.
#
#   e-mail : jimutbahanpal@yahoo.com
#   website : https://jimut123.github.io
#   Created for the purpose of final year project! :=> Almost data visualization project!
#
#   Dated : 10-02-2019
"""

```

```
__version__ = "0.0.8-beta"
__author__ = "Jimut Bahan Pal <jimutbahanpal@yahoo.com>"

from tkinter import Tk, Label, Button, Entry, StringVar, DISABLED, NORMAL, END, W, E, N, S
# tranforming json file into a pandas dataframe library
from http.server import BaseHTTPRequestHandler, HTTPServer
from pandas.io.json import json_normalize
from folium.plugins import MarkerCluster
from tempfile import NamedTemporaryFile
from geopy.geocoders import Nominatim # module to convert an address into latitude and longitude values
from IPython.core.display import HTML
from IPython.display import Image
from datetime import datetime
from tkinter import *
import tkinter as tk
import pandas as pd # library for data analsysis
import numpy as np # library to handle data in a vectorized manner
import subprocess
import webbrowser
import requests # library to handle requests
import argparse
import random # library for random number generation
import folium # plotting library
import json
import os

"""
The arguments that are provided to the application
"""

parser = argparse.ArgumentParser()
parser.add_argument("-q","--quiet", help="will not display anything in the terminal",
                    action="store_true")
parser.add_argument("-t","--theme", help="to select theme [1] classic [2] light [3] dark",choices=[1, 2, 3],
                    type=int)
parser.add_argument("-v","-V","--version", help="displays version",
                    action="store_true")
args = parser.parse_args()

if args.version:
    print("version 0.0.8-beta Jimut (TM)")
else:
    if not args.quiet:
        # the starting of the application dialogue
```

```
66.         print('Starting application ... Necessary libraries imported. \n\n JI
    MUT (TM)')
67.
68.     """
69.         The 3 themes for WISP application.
70.     """
71.     if args.theme == None or args.theme == 1:
72.         # print("USING DEFAULT THEME")
73.         # defining JIMUT's classic theme for wisp :
74.         color_bg_app = "#ffd700"
75.         color_msg = "#ffd700"
76.         color_msg_fg = "#000000"
77.         color_dropdown_fg = "#000000"
78.         color_dropdown = "#cfc611"
79.         color_savemap_label = "#ffd700"
80.         color_savemap_label_fg = "#000000"
81.         color_label_select_map_fg = "#000000"
82.         color_label_select_map = "#ffd700"
83.         color_save_map_entry = "#ffffff"
84.         color_save_map_entry_fg = "#000000"
85.         color_entry_default = "#ffffff"
86.         color_entry_default_fg = "#000000"
87.         color_use_def_sec_button = "#ffff00"
88.         color_use_def_sec_button_fg = "#000000"
89.         color_submit_button = "#ffff00"
90.         color_submit_button_fg = "#000000"
91.         color_preference_label = "#ffd700"
92.         color_preference_label_fg = "#000000"
93.         color_preference_canvas = "#ffd700"
94.         color_preference_entry = "#ffffff"
95.         color_preference_entry_fg = "#000000"
96.         color_show_map_button = "#ffff00"
97.         color_show_map_button_fg = "#000000"
98.         color_pref_scrollbar = "#8b4513"
99.
100.            elif args.theme == 2:
101.                #print("THEME : ",args.theme)
102.                # defining JIMUT's light theme for wisp:
103.                color_bg_app = "#7fffd4"
104.                color_msg = "#7fffd4"
105.                color_msg_fg = "#000000"
106.                color_dropdown_fg = "#000000"
107.                color_dropdown = "#7dcea0"
108.                color_savemap_label = "#7fffd4"
109.                color_savemap_label_fg = "#000000"
110.                color_label_select_map_fg = "#000000"
111.                color_label_select_map = "#7fffd4"
112.                color_save_map_entry = "#a1caf1"
113.                color_save_map_entry_fg = "#000000"
114.                color_entry_default = "#a1caf1"
```

```
115.                color_entry_default_fg = "#000000"
116.                color_use_def_sec_button = "#88d8c0"
117.                color_use_def_sec_button_fg = "#000000"
118.                color_submit_button = "#00cc99"
119.                color_submit_button_fg = "#000000"
120.                color_preference_label = "#afe4de"
121.                color_preference_label_fg = "#000000"
122.                color_preference_canvas = "#afe4de"
123.                color_preference_entry = "#bcd4e6"
124.                color_preference_entry_fg = "#000000"
125.                color_show_map_button = "#00fa9a"
126.                color_show_map_button_fg = "#000000"
127.                color_pref_scrollbar = "#50c878"
128.
129.            elif args.theme == 3:
130.                # defining JIMUT's dark theme for wisp:
131.                color_bg_app = "#253529"
132.                color_msg = "#253529"
133.                color_msg_fg = "#fefdfa"
134.                color_dropdown_fg = "#fefdfa"
135.                color_dropdown = "#07853b"
136.                color_savemap_label = "#253529"
137.                color_savemap_label_fg = "#fefdfa"
138.                color_label_select_map_fg = "#fefdfa"
139.                color_label_select_map = "#253529"
140.                color_save_map_entry = "#000000"
141.                color_save_map_entry_fg = "#fefdfa"
142.                color_entry_default = "#000000"
143.                color_entry_default_fg = "#fefdfa"
144.                color_use_def_sec_button = "#0a1195"
145.                color_use_def_sec_button_fg = "#fefdfa"
146.                color_submit_button = "#0a1195"
147.                color_submit_button_fg = "#fefdfa"
148.                color_preference_label = "#1e4d2b"
149.                color_preference_label_fg = "#fefdfa"
150.                color_preference_canvas = "#1e4d2b"
151.                color_preference_entry = "#000000"
152.                color_preference_entry_fg = "#fefdfa"
153.                color_show_map_button = "#0a1195"
154.                color_show_map_button_fg = "#fefdfa"
155.                color_pref_scrollbar = "#50c878"
156.
157.
158.            def_sec_dummy = 0
159.
160.            # utils function for CLI
161.
162.            def get_json_secrets():
163.                # this reads the secrets from the secret.txt file and returns t
    hem in tuple format!
```

```
164.                try:
165.                    with open('secrets.txt', 'r') as f:
166.                        array = json.load(f)
167.                    if not args.quiet:
168.                        print(array)
169.                    # returns a tuple containing client id and client secret
170.                    return str(array['client_id']),str(array['client_secret'])
171.                except:
172.                    if not args.quiet:
173.                        print("NO SECRETS PRESENT, please enter it in te
    xt file secrets.txt ...\n Else use the GUI to input secrets!")
174.
175.
176.            # returns the current time
177.            def time_now():
178.                format = "1;32;40"
179.                s1 = ''
180.                time_stmp = datetime.now().isoformat(timespec='seconds'
    )
181.                s1 += '\x1b[%sm %s \x1b[0m' % (format, time_stmp)
182.                if not args.quiet:
183.                    print("running app : {} ".format(s1),end="")
184.
185.
186.            """
187.            The banner theme for WISP application
188.            """
189.            def banner_wisp():
190.                format = "1;33;40"
191.                s1 = ''
192.
193.                banner = """
194.
195.
196.
197.
198.
199.                                                    0.0.8-beta
200.                                    JIMUT(TM)
201.                    """
202.                s1 += '\x1b[%sm %s \x1b[0m' % (format, banner)
203.                if not args.quiet:
204.                    print(s1)
205.
206.            """
207.            The main class for the WISP application.
208.            Contains the GUI and the Custom made HTTP map generato
    r as different modules.
209.            """
210.            class guiProj:
```

```
211.            """
212.            The actual app class
213.            """
214.            def __init__(self, master):
215.                """
216.                The constructor for creating the GUI of the app using tk
    inter!
217.                """
218.                banner_wisp()
219.                # this is creating the padding for the input/label text etc.

220.                for i in range(100):
221.                    master.columnconfigure(i, pad=3)
222.                    master.rowconfigure(i, pad=3)
223.                self.master = master
224.                master.title("WISP")
225.                # probably the do-able geometry
226.                master.geometry("430x460")
227.
228.                # Shortened version of the code!
229.                """
230.                This part creates the GUI for the upper labels and text b
    ox of the GUI
231.                """
232.                msg_s = ["    CLIENT ID    "," FOURSQUARE SEC
    RET    "," LOC/CITY    "," RADIUS (in meters) >= 1000    "," N
    O. OF PREFERENCE    "]
233.                i_var = 0
234.
235.                # just initializing!
236.                self.msg_list = [None]*int(5)
237.                self.text_list = [None]*int(5)
238.                self.label_list = [None]*int(5)
239.                self.entry_list = [None]*int(5)
240.
241.                # creating the basic template of the application!
242.                for msg in msg_s:
243.                    self.msg_list[i_var] = msg
244.                    self.text_list[i_var] = StringVar()
245.                    self.text_list[i_var].set("{}".format(self.msg_list[i_va
    r]))
246.                    self.label_list[i_var] = Label(master, textvariable=sel
    f.text_list[i_var],background=color_msg,foreground=color_msg_fg)
247.                    self.label_list[i_var].grid(row=i_var, column=0, colu
    mnspan=1, sticky=W+E)
248.
249.                    self.entry_list[i_var] = Entry(master,background=col
    or_entry_default,foreground=color_entry_default_fg)
250.                    self.entry_list[i_var].grid(row=i_var, column=1, colu
    mnspan=1, sticky=W+E)
```

```
251.
252.                     i_var += 1
253.
254.                 def submit_pref():
255.                     # this function gets called when they submit the prefe
    rence!
256.                     get_pref_no = self.entry_list[4].get()  #to get the pref
    erence
257.                     # dummy preference for conditional check later
258.                     if get_pref_no == "":
259.                         get_pref_no = 0
260.                     # initialising etc.
261.                     self.entry_pref = [None]*int(get_pref_no)
262.                     self.text_pref = [None]*int(get_pref_no)
263.                     self.pf_text = [None]*int(get_pref_no)
264.                     self.label_pref = [None]*int(get_pref_no)
265.
266.
267.                     def onFrameConfigure(canvas):
268.                         '''Reset the scroll region to encompass the inner fr
    ame'''
269.                         canvas.configure(scrollregion=canvas.bbox("all"))

270.
271.                     def populate(frame):
272.                         '''Put in some fake data'''
273.                         # automating the boring and tedious stuffs through
    list
274.                         # basically, takes the input for the number of prefer
    ences!
275.                         for iter_ in range(int(get_pref_no)):
276.
277.                             text_str = "{}-
    {} :".format("preference",iter_+1)
278.
279.                             self.pf_text[iter_] = text_str
280.                             # creating the label
281.                             self.text_pref[iter_] = StringVar()
282.                             self.text_pref[iter_].set(self.pf_text[iter_])
283.                             self.label_pref[iter_] = Label(frame, textvariable
    =self.text_pref[iter_],background=color_preference_label,foreground=
    color_preference_label_fg)
284.                             self.label_pref[iter_].grid(row=5+iter_+1, colum
    n=0, sticky=W+N)
285.
286.                             # entry widget
287.                             self.entry_pref[iter_] = Entry(frame,background
    =color_preference_entry,foreground=color_preference_entry_fg)
288.                             self.entry_pref[iter_].grid(row=5+iter_+1, colu
    mn=1, sticky=W+N)
```

```
289.
290.                 """
291.                 This part creates the GUI for the upper labels and text
     box of the GUI
292.                 """
293.
294.                 canvas = tk.Canvas(master, borderwidth=0,backgrou
    nd=color_preference_canvas)
295.
296.                 canvas.config(width=250, height=200)
297.
298.                 frame = tk.Frame(canvas,background=color_preferen
    ce_canvas)
299.                 vsb = tk.Scrollbar(master, orient="vertical", comman
    d=canvas.yview, background=color_pref_scrollbar)
300.                 canvas.configure(yscrollcommand=vsb.set)
301.
302.                 # for the scrollbar
303.                 vsb.grid(row=8, column=1,rowspan=int(get_pref_no)
    , sticky="nsw")
304.
305.                 # for the grid
306.                 canvas.grid(row=8,column=0,rowspan=1,sticky="nse
    w")
307.                 canvas.create_window((4,4), window=frame, anchor
    ="nw")
308.                 frame.bind("<Configure>", lambda event, canvas=ca
    nvas: onFrameConfigure(canvas))
309.                 populate(frame)
310.
311.                 # disabling the button! for one-time use!
312.                 self.submit_pref_buttton.configure(state=DISABLE
    D)
313.
314.                 # making different types of maps for more feature ric
    h visualizations
315.                 MAP_TYPES = ["Mapbox Bright","Stamen Toner","
    Stamen Terrain","OpenStreetMap","Mapbox Control Room"]
316.
317.                 self.label_select_map = Label(master, text="SELEC
    T MAP-
    TYPE",foreground=color_label_select_map_fg,background=color_lab
    el_select_map)
318.                 self.label_select_map.grid(row=int(get_pref_no)+8,c
    olumn=0,columnspan=1,sticky=W+E+N+S)
319.                 self.dropdown_map_select = StringVar(master)
320.                 self.dropdown_map_select.set(MAP_TYPES[0])
321.
322.                 # foreground=color_dropdown_fg,background=color
    _dropdown,
```

```
323.                    self.dropdown_menu = OptionMenu(master,self.drop
    down_map_select,*MAP_TYPES)
324.                    # to set the color of the dropdown menu to a different
     color
325.                    self.dropdown_menu.config(foreground=color_dropd
    own_fg,background=color_dropdown)
326.
327.                    self.dropdown_menu.grid(row=int(get_pref_no)+8,co
    lumn=1,columnspan=1,sticky=W+E+N+S)
328.                    # use dropdown_map_select.get() to get the contents
    of this list
329.
330.                    # now save the file's entry
331.
332.                    self.save_map = Label(master, text="SAVE MAP AS
     (OPTIONAL)",background=color_savemap_label,foreground=color_s
    avemap_label_fg)
333.                    self.save_map.grid(row=int(get_pref_no)+9,column=
    0,columnspan=1,sticky=W+E+N+S)
334.
335.                    # for the save map entry file
336.                    self.save_map_entry = Entry(master,background=col
    or_save_map_entry,foreground=color_save_map_entry_fg)
337.                    self.save_map_entry.grid(row=int(get_pref_no)+9,co
    lumn=1,columnspan=1, sticky=W+E)
338.
339.                    self.show_map_button = Button(master, text="show
    map",command=self.show_map,background=color_show_map_button
    ,foreground=color_show_map_button_fg)
340.                    self.show_map_button.grid(row=int(get_pref_no)+10
    ,column=1,columnspan=1, sticky=W+E)
341.
342.
343.                def def_sec():
344.                    # this function sets the default secrets by reading the f
    ile secrets.txt, the secrets are stored as JSON
345.
346.                    global def_sec_dummy
347.                    try:
348.                        test1,test2 =get_json_secrets()
349.                        def_sec_dummy = 1
350.                        time_now()
351.                        if not args.quiet:
352.                            print("USING DEFAULT SECRETS FOR CLI
    ENT_ID and CLIENT_SECRET ")
353.                        self.entry_list[0].insert(END, 'using default client I
    D')
354.                        self.entry_list[1].insert(END, 'using default client s
    ecret')
355.                        # disabling things! lol
```

```
356.                     self.use_default_sec.configure(state=DISABLED)

357.                 except:
358.                     def_sec_dummy = 0
359.                     time_now()
360.                     if not args.quiet:
361.                         print("PLEASE ENTER SECRETS IN secrets.tx
    t file in the current directory!")
362.                     time_now()
363.                     if not args.quiet:
364.                         print("QUITTING")
365.                     exit(4)
366.
367.                 # place holder gets called when we use this! (default sec
    ret thingie)
368.                 self.use_default_sec = Button(master, text="use default
    secrets",command=def_sec,background=color_use_def_sec_button,for
    eground=color_use_def_sec_button_fg)
369.                 self.use_default_sec.grid(row=5,column=0, sticky=W+
    E)
370.                 # again button thing
371.                 self.submit_pref_buttton = Button(master, text="submit
    ",command=submit_pref,background=color_submit_button,foreground
    =color_submit_button_fg)
372.                 self.submit_pref_buttton.grid(row=5,column=1,column
    span=1, sticky=W+E)
373.
374.
375.             def show_map(self):
376.                 # fetching the name of the map to be save here too!
377.                 time_now()
378.                 global save_name_map
379.                 try:
380.                     save_name_map = str(self.save_map_entry.get())
381.                     time_now()
382.                     if save_name_map == "":
383.                         if not args.quiet:
384.                             time_now()
385.                             print("NO FILE NAME GIVEN!..\n SO NOT S
    AVING!")
386.                     else:
387.                         if not args.quiet:
388.                             print("FILE NAME GOT !!: ",save_name_map)

389.                 except:
390.                     if not args.quiet:
391.                         time_now()
392.                         print("NO FILE NAME GIVEN!..\n SO NOT SA
    VING!")
393.                     save_name_map=None
```

```
394.
395.                 # To get all the values and show the map!
396.                 all_values = []     # has all the values that is got from the
    GUI
397.                 for item in self.entry_list:
398.                     all_values.append(item.get())
399.                 pref_list = []      # gets the preference one by one!
400.                 for item in self.entry_pref:
401.                     pref_list.append(item.get())
402.                 #print(all_values)
403.                 string_gen = "0123456789abcdef"     # this actually gen
    erated the random hex code for colors!
404.                 def get_random_col():
405.                     # unnecessary stuffs to make the visualization cool
406.                     ret_str = "#"
407.                     for i in range(6):
408.                         ret_str += random.choice(string_gen)
409.                     return ret_str
410.                 #get_random_col()
411.                 global def_sec_dummy
412.                 if def_sec_dummy == 1:
413.                     # default things! lol make sure to clear them!!!!
414.                     # this takes the tuple returned from the get_json_secr
    ets() function which reads secrets from secrets.txt file
415.                     time_now()
416.                     CLIENT_ID, CLIENT_SECRET = get_json_secrets(
    )
417.
418.
419.                 elif def_sec_dummy == 0:
420.                     CLIENT_ID = all_values[0]                #input("Enter
     the client ID : ") # your Foursquare ID
421.                     CLIENT_SECRET = all_values[1]             #input("
    Enter the Foursquare secret : ") # your Foursquare Secret
422.
423.                 VERSION = '20190122'
424.                 LIMIT = 1000
425.                 address = all_values[2]                          #input("En
    ter the location/ city :")
426.                 time_now()
427.                 if not args.quiet:
428.                     print('Your credentails:')
429.                     time_now()
430.                     print('CLIENT_ID: ' + CLIENT_ID)
431.                     time_now()
432.                     print('CLIENT_SECRET:' + CLIENT_SECRET)
433.                     time_now()
434.                     print('Location of your choice : ', address)
435.
436.                 geolocator = Nominatim(timeout = 10)
```

```
437.                 try:
438.                     # get's the lat and long for a place, it is kept under try
    catch for safety purpose
439.                     location = geolocator.geocode(address)
440.                     latitude = location.latitude
441.                     longitude = location.longitude
442.                     if not args.quiet:
443.                         time_now()
444.                         print(latitude, longitude)
445.                 except:
446.                     if not args.quiet:
447.                         time_now()
448.                         print("CHECK INTERNET CONNECTION!\n EL
    SE YOUR NET IS NOT IN FULL 3/4G")
449.                     # directly closes the application
450.                     exit(4)
451.
452.
453.                 RADIUS = int(all_values[3])
454.                 if not args.quiet:
455.                     time_now()
456.                     print("Total preference list : ",pref_list)
457.
458.                 # To clean the list if by chance someone has given unne
    cessary values or empty values or unused text entry box
459.                 pref_list = list(filter(None, pref_list))
460.                 if not args.quiet:
461.                     time_now()
462.                     print("New pref list : ",pref_list)
463.
464.                 map_address = folium.Map(location=[latitude, longitud
    e], zoom_start=11)
465.                 marker_cluster = MarkerCluster().add_to(map_address)

466.                 list_df = []
467.
468.
469.                 col_fill = []
470.                 col_border = []
471.                 for item_pref in pref_list:
472.                     url = 'https://api.foursquare.com/v2/venues/search?cli
    ent_id={}&client_secret={}&ll={},{}&v={}&query={}&radius={}&
    limit={}'.format(CLIENT_ID, CLIENT_SECRET, latitude, longitude,
    VERSION, item_pref, RADIUS, LIMIT)
473.                     try :
474.                         if not args.quiet:
475.                             time_now()
476.                             print("url : ",url)
477.                         results = requests.get(url).json()
478.                         # assign relevant part of JSON to venues
```

```
                venues = results['response']['venues']

                # tranform venues into a dataframe
                dataframe = json_normalize(venues)
                if not args.quiet:
                    time_now()
                    print(dataframe.head())
                try:
                    # keep only columns that include venue name, and anything that is associated with location
                    filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if col.startswith('location.')] + ['id']
                    dataframe_filtered = dataframe.loc[:, filtered_columns]
                except:
                    if not args.quiet:
                        time_now()
                        print("Something went wrong!")
                    continue
                # function that extracts the category of the venue
                def get_category_type(row):
                    try:
                        categories_list = row['categories']
                    except:
                        categories_list = row['venue.categories']

                    if len(categories_list) == 0:
                        return None
                    else:
                        return categories_list[0]['name']

                # filter the category for each row
                try:
                    dataframe_filtered['categories'] = dataframe_filtered.apply(get_category_type, axis=1)
                except:
                    if not args.quiet:
                        time_now()
                        print("Something went wrong!")
                    continue

                # clean column names by keeping only last term
                dataframe_filtered.columns = [column.split('.')[-1] for column in dataframe_filtered.columns]

                #dataframe_filtered.head()

                # copying it to stationary shop dataframe

                data_frame = dataframe_filtered.copy()
```

```
                        list_df.append(data_frame)
                    except:
                        if not args.quiet:
                            time_now()
                            print("Preference : ",item_pref," doesn't exists!!!
    ")
                if not args.quiet:
                    time_now()
                    print(list_df)
                # create map latitude and longitude values
                if not args.quiet:
                    time_now()
                    print("MAP SELECTED :=> ",self.dropdown_map_s
    elect.get())

                MAP_FINAL = folium.Map(location=[latitude, longitu
    de], tiles=str(self.dropdown_map_select.get()),zoom_start=11)
                # configuration for the dafault map to be created!
                marker_cluster = MarkerCluster().add_to(MAP_FINAL
    )

                for list_item in list_df:
                    FILL_COL = str(get_random_col())
                    OVER_COL = str(get_random_col())
                    col_fill.append(FILL_COL)
                    col_border.append(OVER_COL)

                    # Check whether the values are actually present or no
    t!
                    # for latitudes
                    try:
                        len_data = len(list_item['lat'])
                        if not args.quiet:
                            time_now()
                            print("TOTAL DATA FETCHED",len_data)
                            time_now()
                            print(list_item['lat'])
                    except:
                        #pass
                        if not args.quiet:
                            print("NOT got any 'lat' values! using default")
                        list_item['lat'] = [None]*int(len_data)

                    # for longitudes
                    try:
                        if not args.quiet:
                            time_now()
                            print("TOTAL DATA FETCHED",len(list_item
    ['lng']))
                            time_now()
```

```
                print(list_item['lng'])
            except:
                #pass
                if not args.quiet:
                    print("NOT got any 'lng' values! using default")

                list_item['lng'] = [None]*int(len_data)
                if not args.quiet:
                    time_now()
                    print(list_item['lng'])

            # for categories

            try:
                if not args.quiet:
                    time_now()
                    print("TOTAL DATA FETCHED",len(list_item
    ['categories']))
                    time_now()
                    print(list_item['categories'])
            except:
                #pass
                if not args.quiet:
                    print("NOT got any 'categories' values! using de
    fault")
                list_item['categories'] = [None]*int(len_data)
                if not args.quiet:
                    time_now()
                    print(list_item['categories'])

            # for postalCode values

            try:
                if not args.quiet:
                    time_now()
                    print("TOTAL DATA FETCHED",len(list_item
    ['postalCode']))
                    time_now()
                    print(list_item['postalCode'])
            except:
                #pass
                if not args.quiet:
                    print("NOT got any 'postalCode' values! using d
    efault")
                list_item['postalCode'] = [None]*int(len_data)
                if not args.quiet:
                    time_now()
                    print(list_item['postalCode'])


```

```
613.                     for lat, lng, cat, postcode in zip(list_item['lat'], list_ite
    m['lng'],  list_item['categories'],  list_item['postalCode']):
614.                         f_format = str("POSTCODE : "+str(postcode))
615.                         s_format = str("CATEGORY : "+str(cat))
616.                         label = '{}, {}'.format(f_format,s_format )
617.                         label = folium.Popup(label, parse_html=True)
618.                         # the circle marker is done through this way! with
    custom pop-ups
619.                         folium.CircleMarker(
620.                             [lat, lng],
621.                             radius=5,
622.                             popup=label,
623.                             color=OVER_COL,
624.                             fill=True,
625.                             fill_color=FILL_COL,
626.                             fill_opacity=0.7).add_to(marker_cluster)
627.                     dots_html = ""
628.                     for var1, var2, var3 in zip(pref_list,col_fill,col_borde
    r):
629.                         dots_html = dots_html + """
630.                           {}
631.                         <svg height="10" width="10">
632.                         <circle cx="5" cy="5" r="4" stroke="{}" stroke-
    width="3" fill="{}" />
633.                         </svg><br/> """.format(str(var1),str(var3),str(var2)
    )
634.
635.                     if not args.quiet:
636.                         time_now()
637.                         print(dots_html)
638.                 legend_html = """
639.                         <div style = "position: fixed; top: 5px; left: 900px;
    z-index:9999;">
640.                             <h1 ><i style="color:#c6ae0d; font face=Verdan
    a,Arial,Helvetica "> WISP </i></h1>
641.                         </div>
642.                         <div style="position: fixed;
643.                         bottom: 50px; left: 50px;
644.                         border:2px solid grey; z-index:9999; font-
    size:14px;">
645.                             <b>Legend <b>
646.                             <br>{}
647.                         </div>
648.                         <div style="position: fixed;
649.                         bottom: 50px; right: 50px;
650.                         z-index:9999; font-size:10px;
651.                         ">
652.                             <b style="color:#f90404" style="align: justified"
    >
653.                                 WISP <br/>
```

```
654.                                 version: 0.0.8-beta <br/>
655.                                 ©Jimut Bahan Pal <br/>
656.                                 Author : jimutbahanpal@yahoo.com
657.                             </b>
658.                         </div>
659.                         """.format(str(dots_html))
660.                 time_now()
661.                 if not args.quiet:
662.                     print(legend_html)
663.
664.                 MAP_FINAL.get_root().html.add_child(folium.Elemen
    t(legend_html))
665.                 # setting port addr, localhost for the custom http server
666.                 PORT = 7000
667.                 HOST = '127.0.0.1'
668.                 SERVER_ADDRESS = '{host}:{port}'.format(host=H
    OST, port=PORT)
669.                 FULL_SERVER_ADDRESS = 'http://' + SERVER_AD
    DRESS
670.                 # ----------------------------------------------------------------
    ------------------------------
671.                 # so let's write a custom temporary-HTML renderer
672.                 def TemproraryHttpServer(page_content_type, raw_dat
    a):
673.                     """
674.                     A simpe, temprorary http web server on the pure Pyth
    on 3.
675.                     It has features for processing pages with a XML or H
    TML content.
676.                     """
677.
678.                     class HTTPServerRequestHandler(BaseHTTPReques
    tHandler):
679.                         """
680.                         An handler of request for the server, hosting XML-
    pages.
681.                         """
682.
683.                         def do_GET(self):
684.                             """Handle GET requests"""
685.
686.                             # response from page
687.                             self.send_response(200)
688.
689.                             # set up headers for pages
690.                             content_type = 'text/{0}'.format(page_content_t
    ype)
691.                             self.send_header('Content-type', content_type)
692.                             self.end_headers()
693.
```

```
694.                         # writing data on a page
695.                         self.wfile.write(bytes(raw_data, encoding='utf'))

696.
697.                         return
698.
699.                     if page_content_type not in ['html', 'xml']:
700.                         raise ValueError('This server can serve only HTM
    L or XML pages.')
701.
702.                     page_content_type = page_content_type
703.
704.                     # kill a process, hosted on a localhost:PORT
705.                     time_now()
706.                     subprocess.call(['fuser', '-
    k', '{0}/tcp'.format(PORT)])
707.
708.                     # Started creating a temprorary http server.
709.                     httpd = HTTPServer((HOST, PORT), HTTPServerR
    equestHandler)
710.
711.                     # run a temprorary http server
712.                     httpd.serve_forever()
713.
714.
715.                 def run_html_server(html_data=None):
716.
717.                     if html_data is None:
718.                         html_data = """
719.                         <!DOCTYPE html>
720.                         <html>
721.                         <head>
722.                         <title>Page Title</title>
723.                         </head>
724.                         <body>
725.                         <h1>This is a Heading</h1>
726.                         <p>This is a paragraph.</p>
727.                         </body>
728.                         </html>
729.                         """
730.
731.                     # open in a browser URL and see a result
732.                     webbrowser.open(FULL_SERVER_ADDRESS)
733.
734.                     # run server
735.                     TemproraryHttpServer('html', html_data)
736.
737.                 # ----------------------------------------------------------
    ------------------------------
```

```
                # now let's save the visualization into the temp file and r
    ender it
                tmp = NamedTemporaryFile()
                MAP_FINAL.save(tmp.name)
                with open(tmp.name) as f:
                    folium_map_html = f.read()
                global root
                root.destroy()
                if not args.quiet:
                    time_now()
                    print("Destroying window!! exiting from GUI to We
    b - Browser")

                # to get the name of the file to be saved!
                if save_name_map == None:
                    pass
                else:
                    name_final = save_name_map+".html"
                    if save_name_map==None:
                        pass
                    else:
                        if not args.quiet:
                            time_now()
                            print("WRITING TO HTML FILE !!!")
                        time_now()
                        with open(name_final, 'w') as file_:
                            file_.write(folium_map_html)
                try:
                    if not args.quiet:
                        time_now()
                    os.remove(".html")
                    if not args.quiet:
                        print("CACHES REMOVED!")
                except:
                    if not args.quiet:
                        time_now()
                        print("...CLEANING CACHES!")

                run_html_server(folium_map_html)

            def main():
                global root
                root = Tk()
                root.configure(background=color_bg_app)
                # initialising the app
                guiProj(root)
                # goes on and on loop for tkinter!
                root.mainloop()

            if __name__ == "__main__":
```

```
786.                main()
```

This code can be also found under my Github account **[1].** This is the source code where all the modules are coalesced into one program with the documentation and comments. The code contains the GUI, the data manipulation part and the custom made folium map generator part. This code can be run with a python wisp.py command. Then we provide the setup code, which is used to make this wisp.py software. The code for the setup.py is given below which can also be found in Github **[1]**:

```
1.  from setuptools import setup
2.  import os
3.  def read(fname):
4.      return open(os.path.join(os.path.dirname(__file__), fname)).read()
5.
6.  if __name__ == '__main__':
7.      setup(
8.          name = 'wisp',
9.          version="0.0.8-beta",
10.         description = 'A preference based location finder app',
11.         author = 'Jimut Bahan Pal',
12.         author_email = 'paljimutbahan@gmail.com',
13.         maintainer = 'Jimut Bahan Pal',
14.         maintainer_email = 'paljimutbahan@gmail.com',
15.         url = '',
16.         license = 'GPLv2+',
17.         platforms = 'Linux',
18.         py_modules = ['wisp'],
19.         entry_points = {
20.             'console_scripts': ['wisp = wisp:main'],
21.         },
22.         include_package_data = True,
23.         install_requires = [
24.             'requests',
25.             'datetime',
26.             'IPython',
27.             'pandas',
28.             'folium',
29.             'geopy',
30.             'numpy',
31.             'wget',
32.             'pip'
33.         ],
34.         keywords = 'Preference, location, wisp, requests, html, json, numpy, pandas, software, tkinter, folium, geopy, lat lon, wget, Ipython, location cluster, KNN, Machine learning, clustering',
35.         classifiers = [
```

```
36.             'Development Status :: 0.0.8 - Beta',
37.             'Environment :: Console',
38.             'Intended Audience :: End Users/Desktop',
39.             'Intended Audience :: System Administrators',
40.             'License :: OSI Approved :: GNU General Public License v2
    or later (GPLv2+)',
41.             'Operating System :: Manjaro',
42.             'Programming Language :: Python :: 3.7.2',
43.             'Topic :: Internet :: WWW/HTTP',
44.             'Topic :: Internet :: WWW/HTTP :: Indexing/Search'
45.             ],
46. )
```

These codes form the basis of all the code that makes the wisp application alive. The code that should be run at the beginning of the installation of the application is the requirements.txt code. It is run with the help of pip install –r requirements.txt under administrator privileges so that python could install the necessary third party libraries that are needed to run this software. All of these libraries are open sourced library. The contents of requirements.txt **[1]** are given below:

```
1. requests==2.21.0
2. pandas==0.24.1
3. folium==0.7.0
4. geopy==1.18.1
5. numpy==1.16.1
6. wget==3.2
7. datetime
8. IPython
9. pip
```

This comes with a GNU General Public License (GPL V3). Since the code is open-sourced, it is licensed under GPL version 3. The license can be found in the repository **[1].** A snapshot of the licence is :

```
1.  GNU GENERAL PUBLIC LICENSE
2.                    Version 3, 29 June 2007
3.
4.  Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
5.  Everyone is permitted to copy and distribute verbatim copies
6. of this license document, but changing it is not allowed.
```

This is a fairly big license, and will take all of the project pages, more can be found by following the above reference.

5.3 Code Efficiency

The output map that is generated for visualisation is given in real time. We have made the Wisp application as much efficient as possible. The first version of the application is shown in **Figure 19** and **Figure 20.**

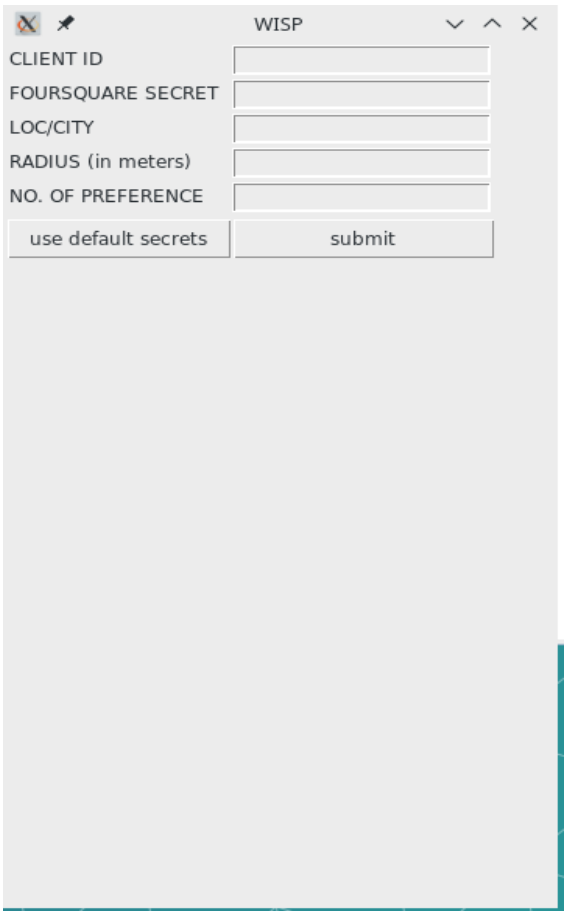

Figure 19: First version of wisp part 1

Figure 20: First version of Wisp part 2

As we can see from the above figures that the application is raw. The first version was made in almost 2 days, using tkinter GUI, and was modified part by part according to the needs.

5.3.1 The actual model of the wisp application

The actual data flow and the improved model of wisp are shown in **Figure 21.** We have used the spiral model for development of Wisp and the final model of the Wisp was developed later, when the requirements and specifications were clear. The diagram makes clear sense of how the Wisp application is working in real time. The output is generated almost in no time after the inputs are given by the user into the application.

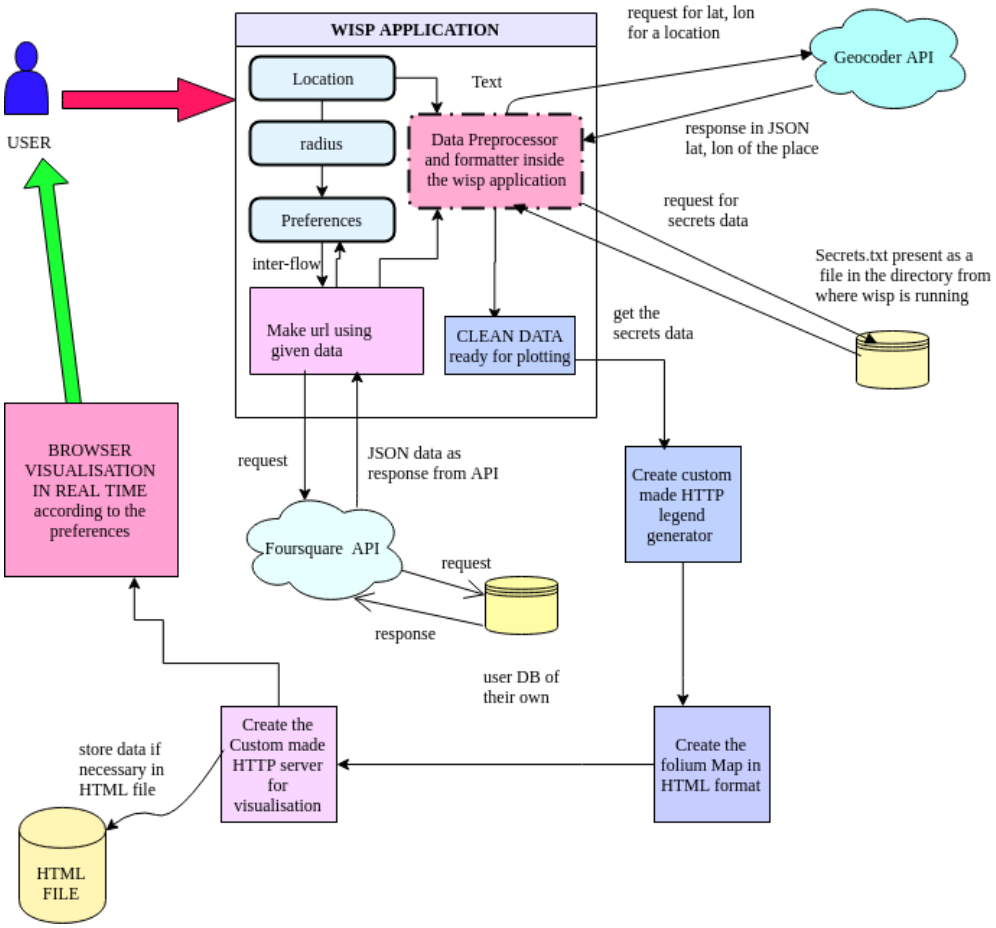

Figure 21: The actual modified logical representation of Wisp application

5.4 Testing Approach

We have done risk based testing and functional testing. Functional testing ensures the usual behaviour of the software **[32]**. Risk based testing is based on software risks. The risk based testing is intended to eliminate the risk involved in the development of the existing software. From a business' point of view, the testing and risk analysis is done in advanced to check the feasibility of the project. It is done to estimate the total cost in advanced. The software

community has done excellent jobs to find bugs early on in the development lifecycle of software. They have done many things to reduce the bugs early often, by providing certain procedures to test the software effectively. The cost of the software testing and fixing bug in the lifecycle of software has been studied by Michael et. al as shown in **Figure 22.**

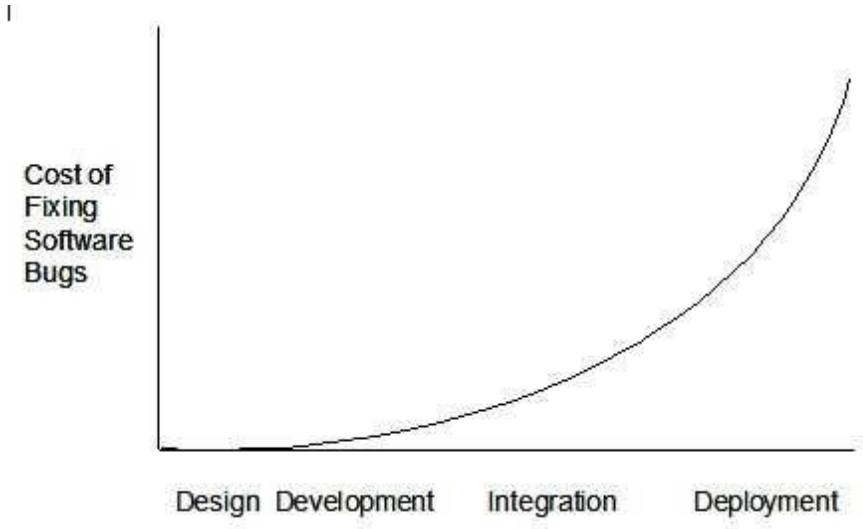

Figure 22: Cost of fixing a bug in different stages of software life cycle **[32]**

Test planning is related to the risk analysis. It starts even when there is no software to test. Test planning is an ongoing process as well, since the requirements and specifications changes over time. The test planning involves more than just series of test. It may also involve test management and automation.

The testing approach is the recipe of how the testing should be carried out in a project. There are two types of testing approaches **[33]**:

- Proactive – the testing approach is carried out as early as possible, in order to fix the defects before the build is created.

- Reactive – an approach where the testing is started after the coding phase.

We have used proactive approaches, since it becomes easy and the prototypes created are always bug free. The strategies that a project can adopt are:

- Dynamic approaches.

- The model based approaches which uses statistical information about the failure rates.

- Risk based testing.
- Methodological approaches.
- The standard industry complaint approaches.

Risk analysis goes on in a development process, it defines the act to mitigate risk, identify risk and higher level management when nothing is possible. It is done with the help of expertise knowledge in the field of risk analysis. The relationship between risk analysis and test planning is shown in **Figure 23.**

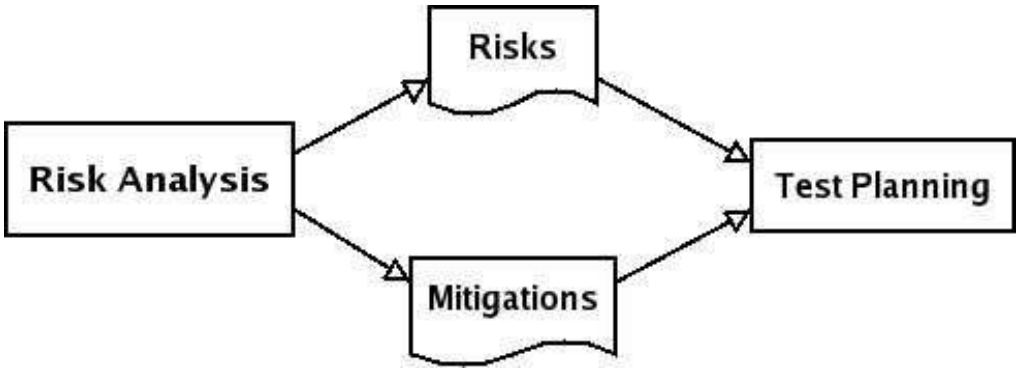

Figure 23: Relationship between risk analysis and test planning **[32]**

So, for the fixing of bugs in the early stages of a software lifecycle, it is necessary to do risk analysis which is proactive, so that each of the prototypes that are built in the process is free from bugs.

5.5 Unit Testing

Unit testing is the first phase of the testing **[32]** that a software application goes through. The "unit" particularly stands for different parts like modules, classes, functions which comprise the software. It is by default functional testing since there is no integration of the modules that is involved. The testing

is done to ensure the tasks that are needed to be done by the software, and how the software is effectively delivering those tasks to the user. The responsibilities of unit testing falls under the shoulders of developers those who can write the tasks and drivers necessary for the test operation to perform.

The benefits of unit testing are **[34]**:

- It increases the confidence of the developers to maintain the code.
- The codes are more reusable, modular, and functional to achieve unit testing.
- The unit testing forms a fast way of building the software since there is no involvement of GUI or extra unnecessary and redundant functionalities, here the test is done on raw codes.
- The cost of unit testing is lower than testing the whole app and debugging the app for hours.
- Codes are more reliable

We have done unit testing by making everything in modules, later we have integrated each of the modules in a program alone. There are a lot of advantages of unit testing. We have followed all the possible ways to make the code reusable and maintainable for future use. There are many tools available for unit testing such as GNU debugger, automated testing in antLR etc, but our application is a simple one and made in parts with the help of modular designs, so we have called the functions, passed the test cases data manually and tested the application for unit testing. We have performed various testing from the beginning, including when the Geocoder API was transferring data to the respective function with the help of internet, and there was a delay in receiving the data, for which the software was not working. Later we found out that it was the timeout factor which was responsible for the delay and the software was coming to a halt.

5.6 Integrated Testing

Integrated testing focuses the testing on a collection of subsystems. There are numerous software bugs that evolve during this testing, because of the way the modules interact with each other. There are security bugs and the traditional ones. The failure to check the input values is one of the most frequent sources of software testing. We have found that there was a time out error in the time of integrated testing for which we were thinking that there might be no data in the API. During the unit testing, it was going right, but during integration testing certain bugs will evolve which doesn't show the error origin. We have implemented a debug mode in the wisp application which can be bypassed by passing an additional argument of –q. This was very helpful in seeing the actual data that was being passed by the functions. The debug mode is shown in the **Figure 24,** where the –q is not passed as an argument.

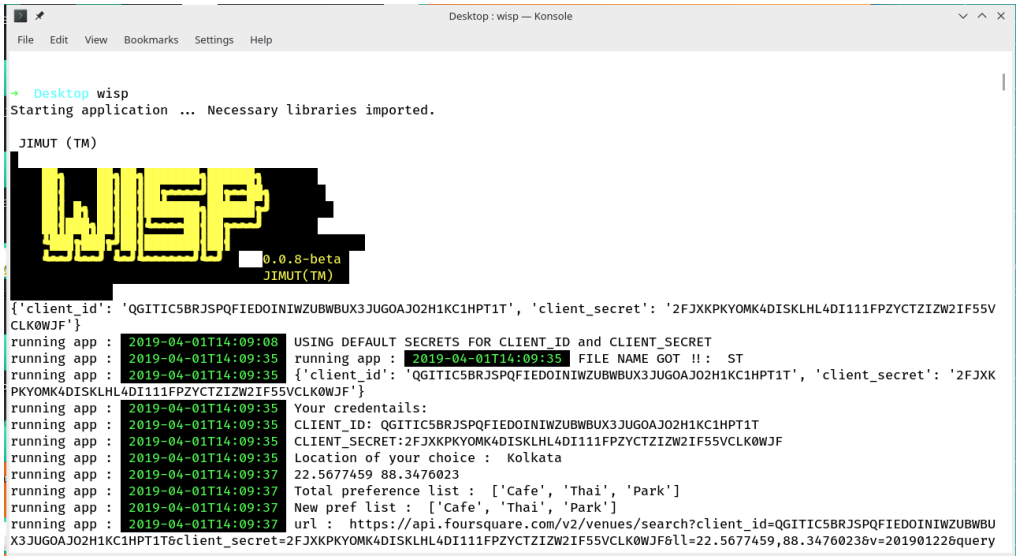

Figure 24: Wisp debug-mode part 1 without passing -q

From the above figure we could see that there is a retro banner that is created on the start of the application. This was actually created by scraping website and making a python module which supports almost 314 fonts. This repository is also open-sourced and is developed by me, can be found by following this link **[36]**. The next part of this image is shown in **Figure 24.** This is the debug mode showing the values or data structures that are passed by different functions of the wisp application. We could see that is taking data and converting it to data frame of the Pandas library.

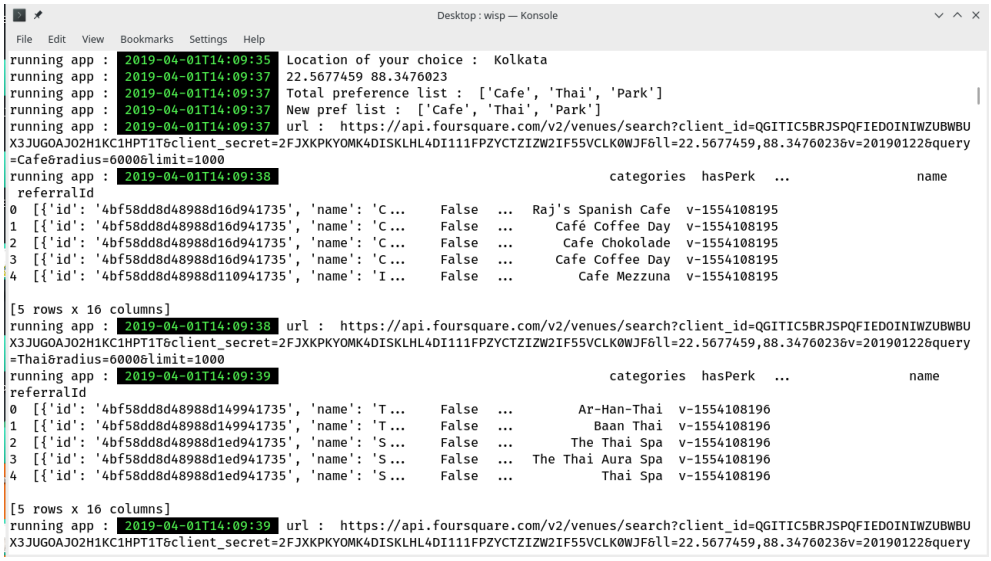

Figure 25: Wisp debug-mode part 2 without passing -q

The working versions of the wisp application is shown, now we will see the dataframe object of the Pandas library that is cleaned and passed to the Folium map in **Figure 26.**

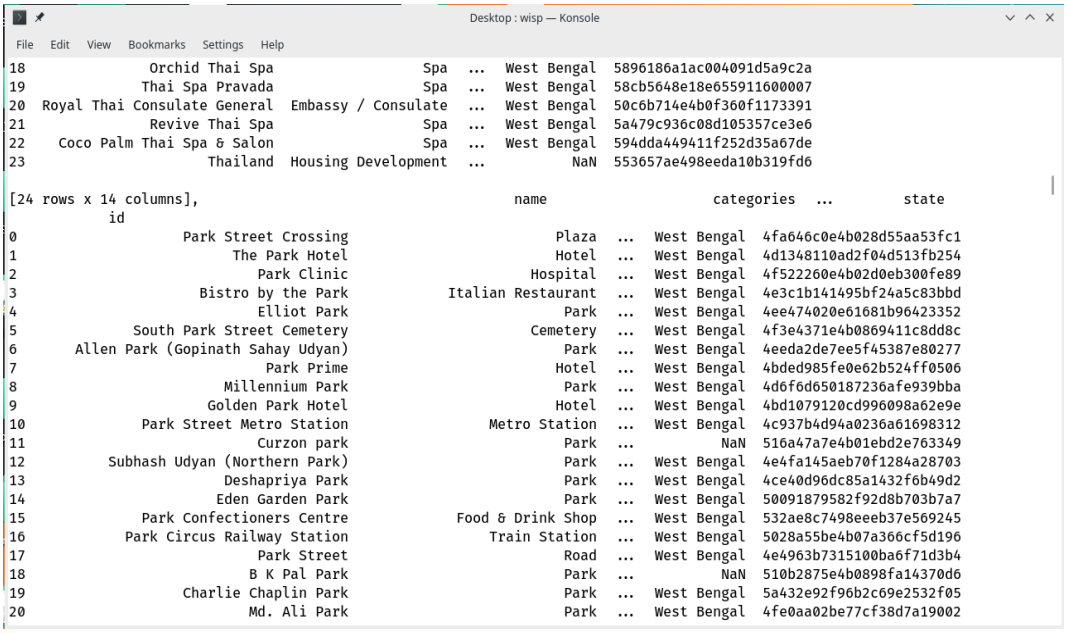

Figure 26: Pandas dataframe object passed to folium map in wisp debug-mode without passing -q

The wisp application's black theme is shown in **Figure 27,** which is got without passing –q as an additional argument. These are shown to ensure the proper working of the wisp application.

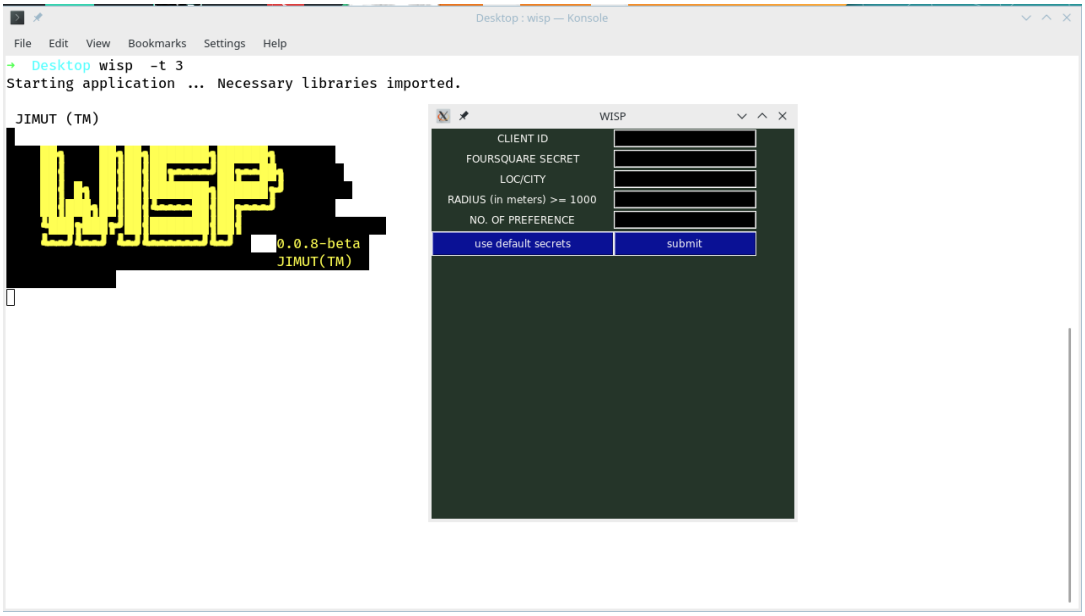

Figure 27: Wisp application's dark theme without -q

The last but not the least, the passing of –q as an optional argument to bypass the debug mode and provide a clean terminal is shown in **Figure 28.**

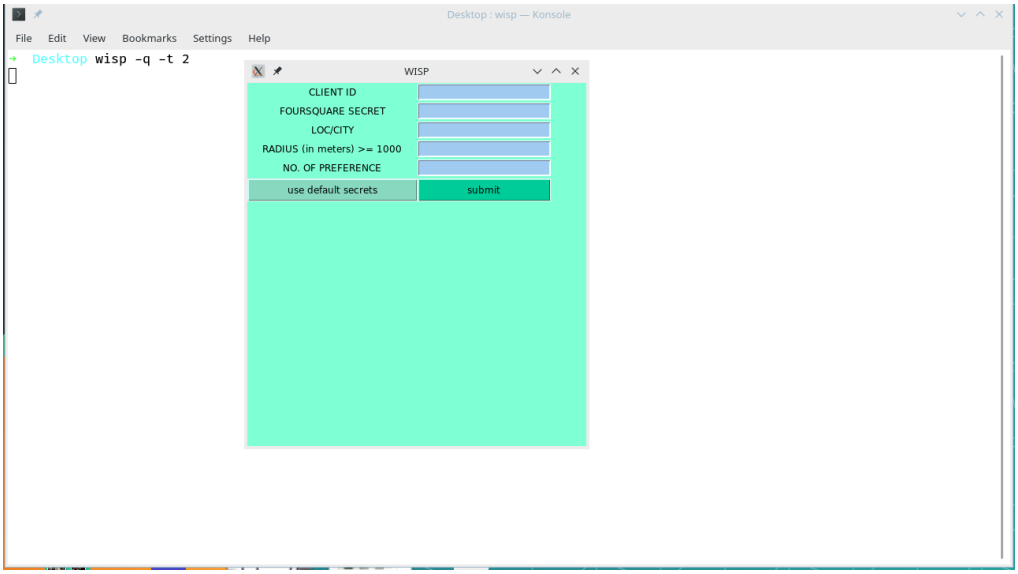

Figure 28: The debug mode is bypassed by passing -q as an argument for clean terminal

This is done to ensure that the terminal remains clean and nothing is displayed on the screen. This is done when the integration testing period finishes.

5.7 Modifications and Improvements

There were modifications and improvements of the software on the whole software development life cycle. We have created many test cases; many parameters were passed until we find it is satisfying. The software was optimised to do real time visualisation. The requirements and specifications were not clear at the beginning but as we developed the software, we grew knowledge about what Python 3 can do and what we can achieve with our available resources. We created the Command Line Interface (CLI) at the beginning and then created the GUI after various test operations.

We performed various changes in the code to ensure its reusability and maintainability. We finally made software which is fully fledged. We used visual studio code. The extra parameters that were generated are shown in **Figure 29.**

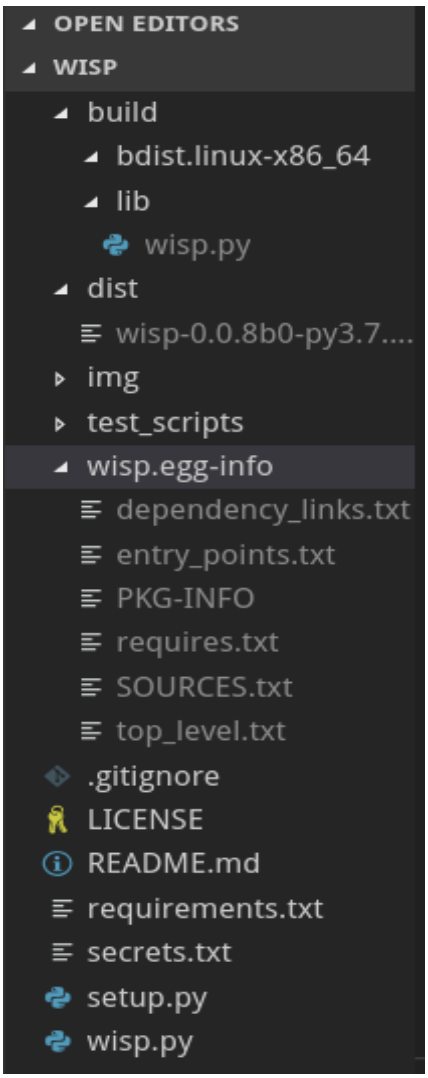

Figure 29: The extra files generated during installation of wisp

From the previous figure we could see that we have only provided setup.py and wisp.py. The additional folders such as build, dist, wisp.egg-info were generated automatically by performing **python setup.py install** under administrative privileges. We have worked in Git based version control environment and all the changes made were committed to a remote server. This ensured that the codes were backed up even if there is any local failure. The first thing that needed to be done while installing this software is to perform **pip install –r requirements.txt** under the root directory of the wisp repository and administrative privileges. The installation of the requirements using the inbuilt pip command in Python is shown in **Figure 30.**

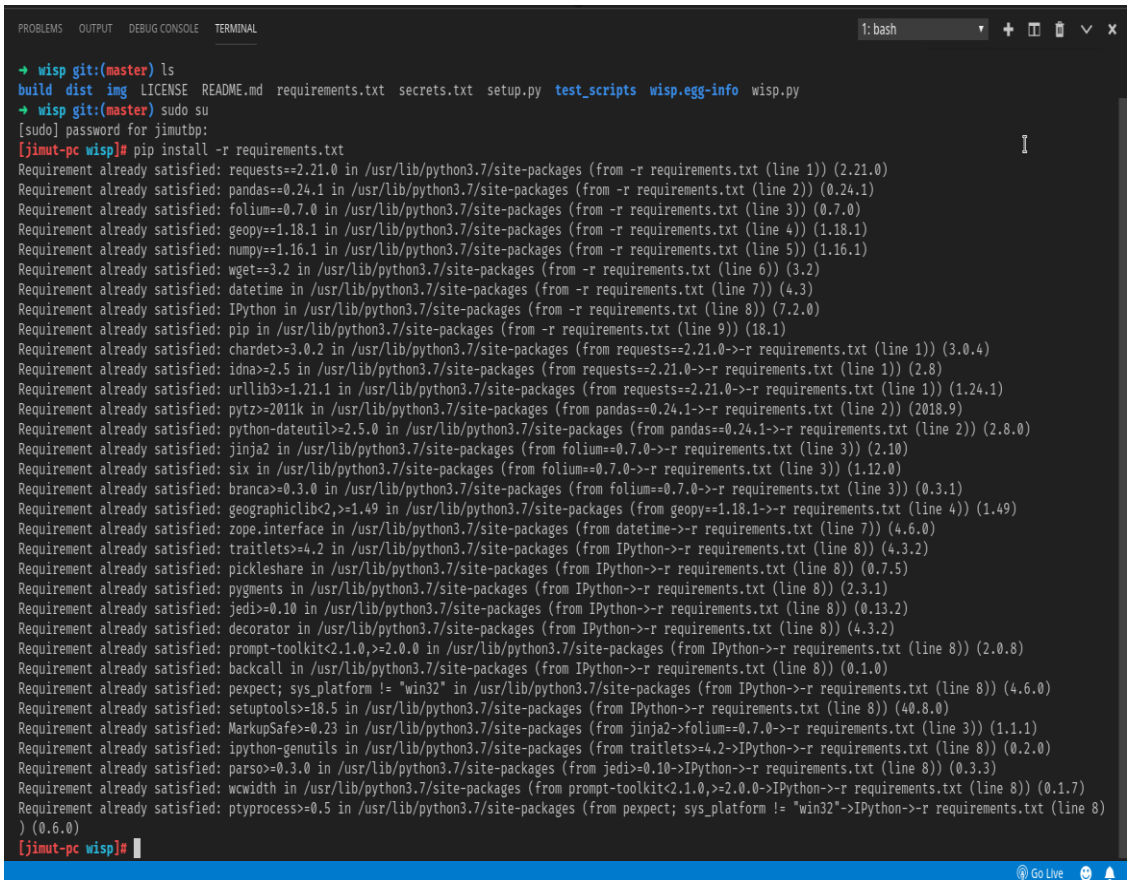

Figure 30: Installation of requirements by pip

The pip command provides a great help when we are installing a lot of modules and external libraries for a project in the machine. The lists of modules that are needed to be inserted are present in a text file one after the other. This helps us to automatically install the requirements every time without manual installation. After the required third party libraries are installed using pip command, we will install the wisp applications software so that we can run the software through the terminal from any directory by just typing the command

wisp. This is done with the help of administrative privileges. A part of the installation of the wisp application is shown in **Figure 31.**

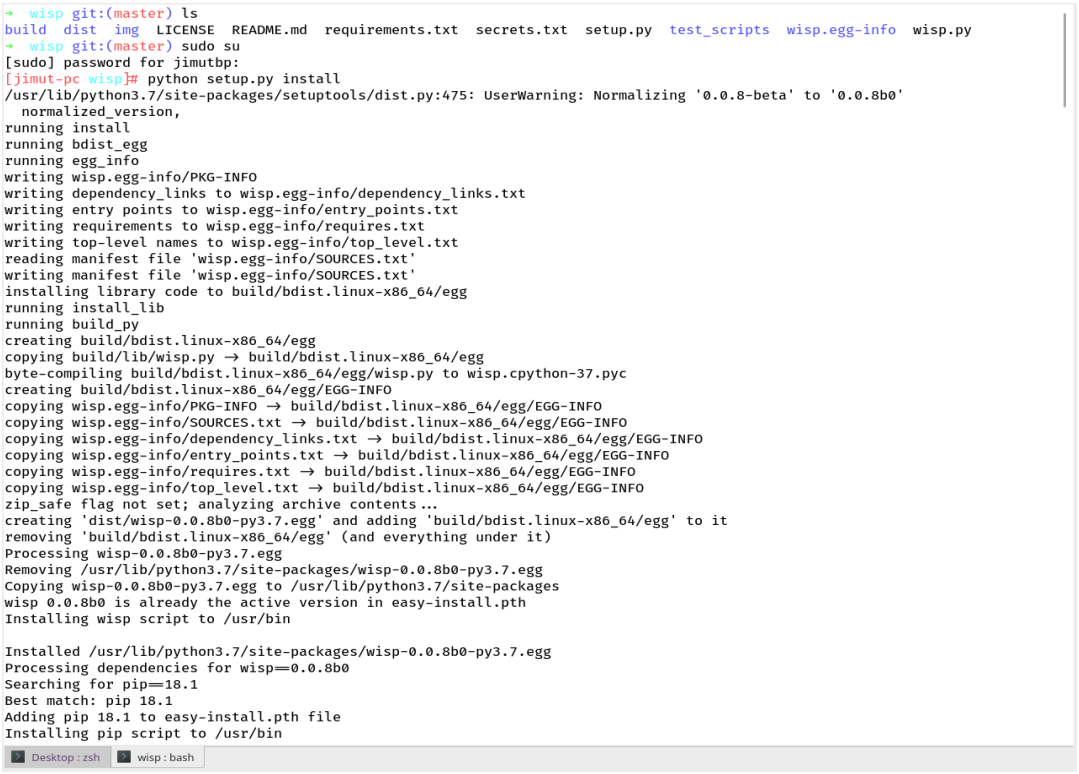

Figure 31: Installation of the wisp application

Here we can see that the wisp application installs itself by typing **python setup.py install,** which is an automated tool for installation of any application software. After this step certain files are generated which will be used by the system to call the wisp application by setting this path in the terminal and system variables.

5.8 Test Reports

We have tested this software for a lot of places; some of the places are shown in this section. **Figure 32** shows the open street map of Pune when four preferences are given. Those preferences include Cafe, Chinese, Park and Hotel. This is the advantage of using FourSquare API, it automatically understands that we are searching for anything relating to Chinese. It may include Chinese food, Chinese restaurant, Chinese museum etc.

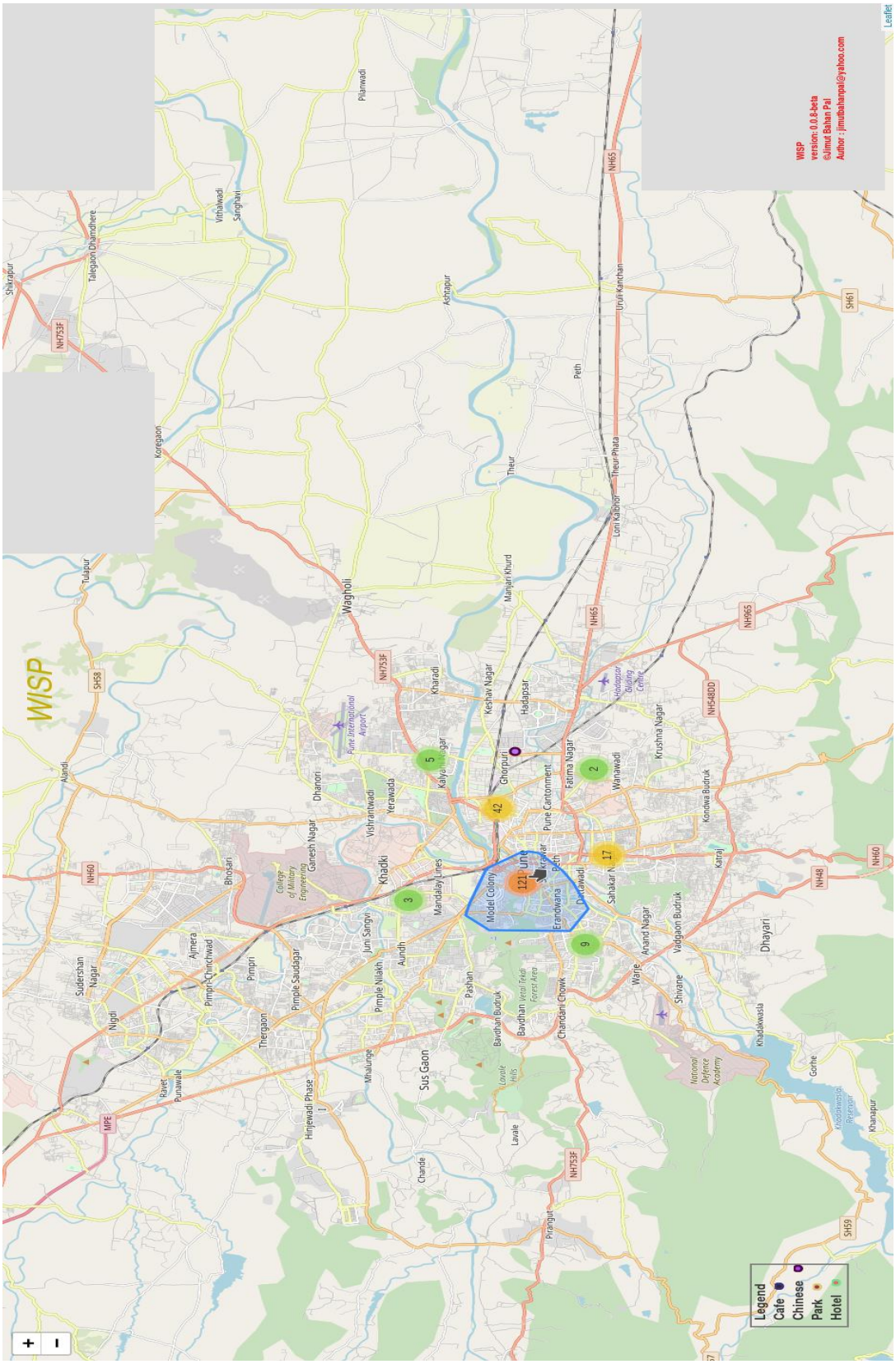

Figure 32: Open Street Map of Pune with four preferences **[36]**

Figure 33: Stamen Toner Map of Kolkata with four preferences **[37]**

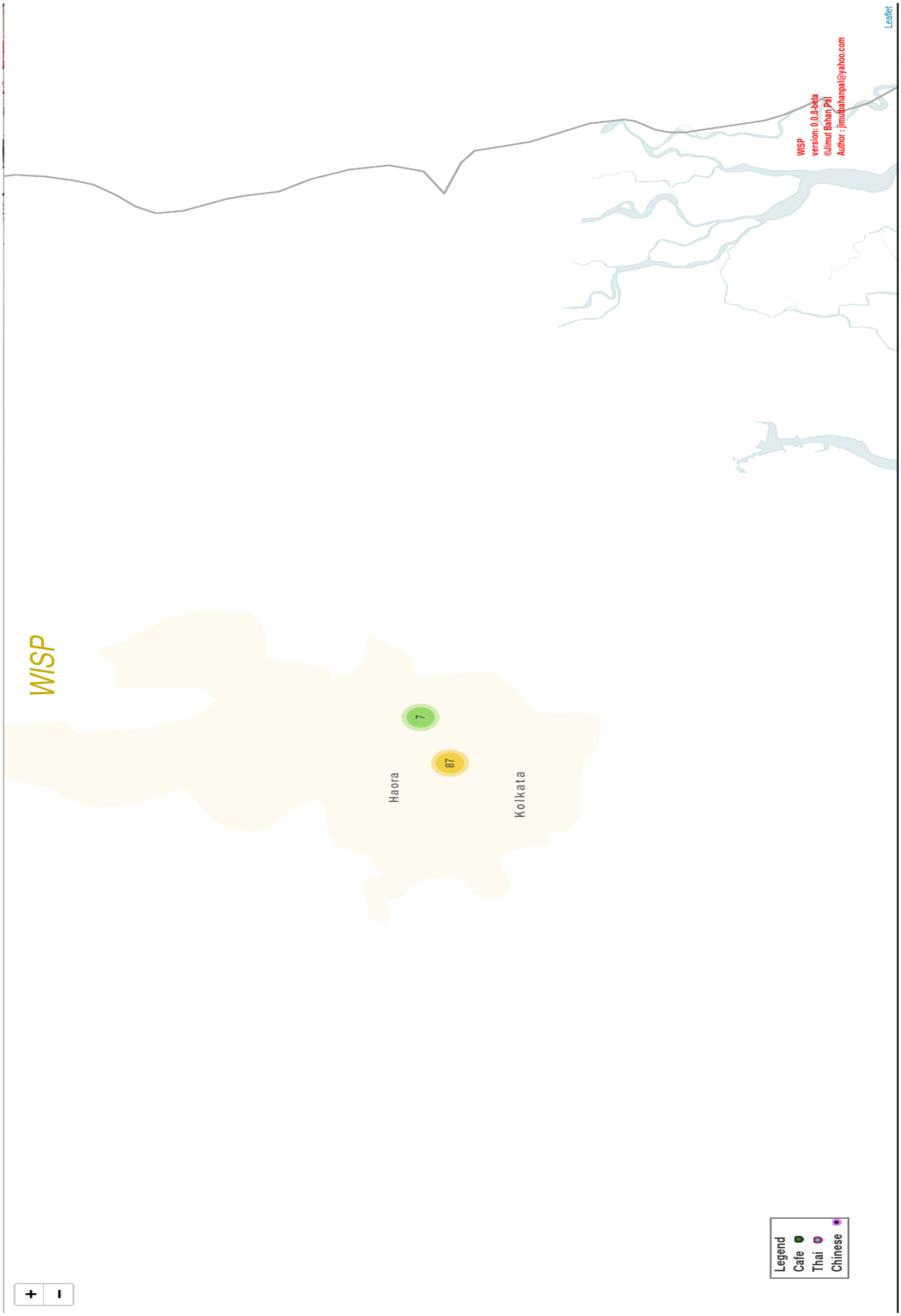

Figure 34: Mapbox Bright Map of Kolkata with 3 preferences **[38]**

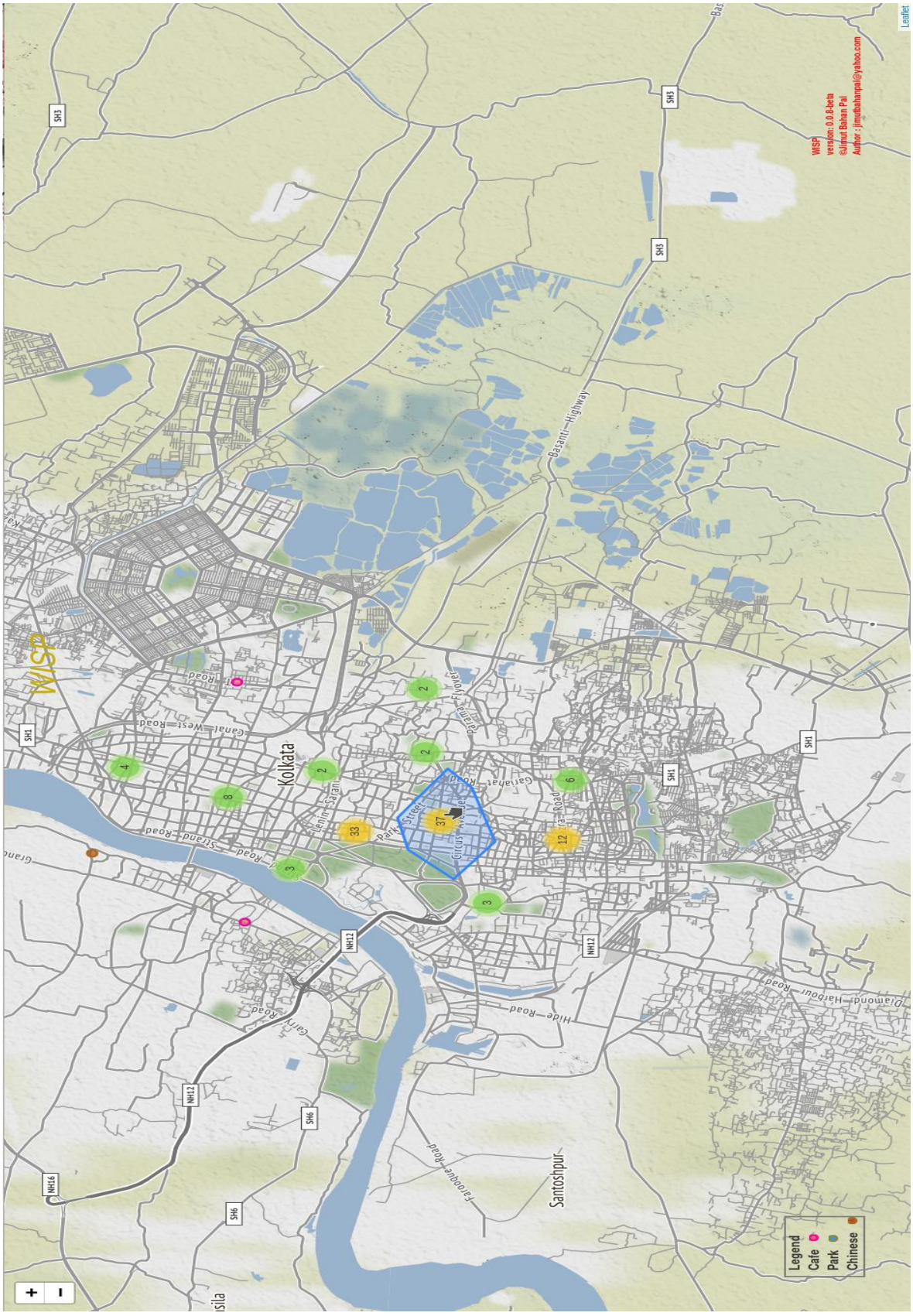

Figure 35: Stamen Terrain map of Kolkata with 3 preferences **[39]**

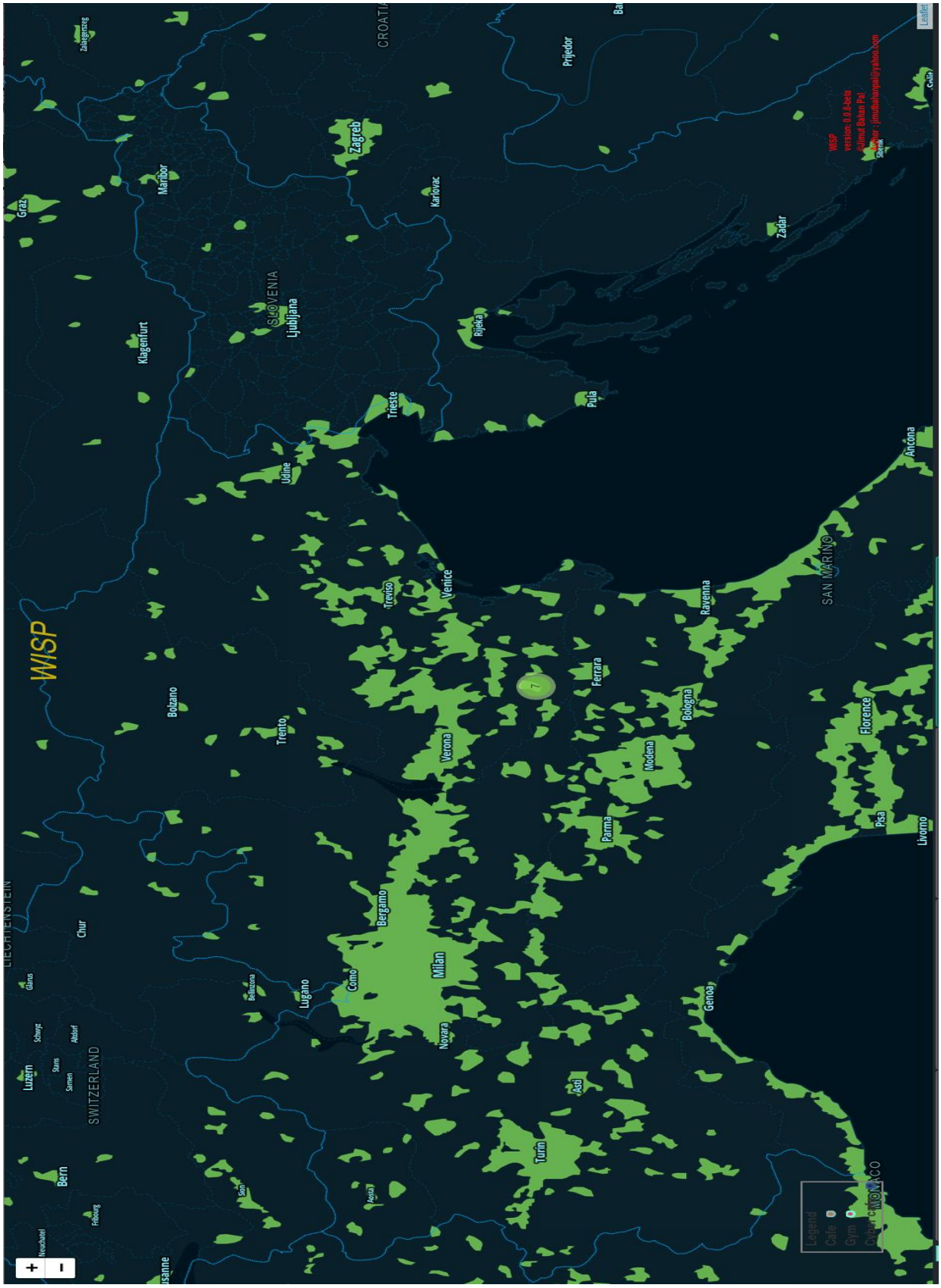

Figure 36: Mapbox Control Room Map of Canada with 3 preferences **[40]**

Figure 33, 34, 35, and 36 consists of all the Maps that are being tested using the Wisp application. The Map is generated and rendered in a browser. The map can also be saved and looked for future use. The reference contains live visual of the maps in the internet. Those are served using cdn-bootstrap in browser from the github repository.

5.9 User Documentation

The documentation is provided in the Github repository **[1]** in README.md. The dissertation documentation is done after all the tedious task of writing the software, testing, integration and release of a stable version. The present version of Wisp is 0.0.8-beta. This is a stable version and can be used by the public. This project is open-sourced and the code is available for free to the public. There is a high chance that this software will be developed in the future by the open source community. The self-documentation is provided in the code. After passing an additional help command as an argument, the documentation can be viewed. The application is pretty much self-intuitive and with procedural generation of user input text box from the GUI. It has a pretty clean UI and can be used easily.

5.10 Conclusion

We have produced a stable version of the app by continuous testing in each version and providing additional features after each stable release. We have seen that by proper planning and creation of the SRS document and the control flow diagrams, the task becomes easy. We could calculate the amount of time that will be necessary to finish the project and the job that will be taken. We also saw that when we divide the task using charts, we provide a deadline, by which we have to give the estimated module to the developer and integrate it for flow of the software.

5.11 Limitations of the system and Future Scope

The main limitation of the system is it uses Markus Cluster to form the clustering part. Which means the quantity of preferences dominates over the group of preferences in total. This means that if we choose Tea, coffee, and suppose Thai food, then it may happen that the Thai food forms a cluster in a different part of the city and the Tea forms enormous cluster in the centre of the city. This means that due to the majority of data points of the Tea, this will form a cluster in the middle of the city whereas we didn't only need Tea; we needed the places which has all the three attributes. So, better algorithm can be built in the future which doesn't bias the cluster according to majority of data points. Another thing that can be done in future is improvement of the GUI. The GUI is self-intuitive and build using tkinter. If it is built using Kivy or anything else, the GUI would have been more appealing. It could be done that the whole system is converted to android application or web application, which will attract a vast range of users. This can be done by using porting tools which can generate the code to be converted for a particular application.

CHAPTER 6: RESULTS AND DISCUSSION

We have built a self-intuitive, easy to use application for common users which will help them to visualise a place and know more about a place by just passing the preferences. We have used professional software developing strategies and tools for the building of this application. We have seen that it becomes comparatively easy when the sketch of the data flow diagrams, scheduling charts, responsibility division are carried out at the beginning of the project. We have also seen that the testing is an important phase in the design of software. Testing by using unit testing tools can be of great importance. This will help to minimise cost and will help to debug small pieces of code with ease.

We have built a fully-fledged application which is comparable to any other small application in Linux. We have provided many features in the application and made it self-intuitive. The main advantage of this application is it is open sourced and will help in future development of the application itself. It is a pleasure to share the code with the open source community with a GPL V3 license. As I have designed this software from scratch, I hope this will be used by a lot of people in the future and will become a different thing by the contributions of more open sourced developers.

CHAPTER 7: CONCLUSIONS

Building software is not enough. Software is known by the number of users that are benefitted by it. Linux is open sourced software. It would have been almost impossible for users to buy software for 10,000 bucks and then throw it away after 5 years when it becomes obsolete. That's why Linus Torvalds created Linux. Linux is another level of standards. It is built essentially for developers. We have built this Wisp application software in Linux. The spirit of sharing code for the benefit of other has been given by Richard Stallman. By open-sourcing code we give the software a new life. Linux was created by Torvalds and he has written about 2% of the total code in it. The main aim of open sourcing the code is to give it a new life. It may be a summer project to develop the Wisp application in terms of algorithms, in terms of GUI etc.

The satisfaction of building a working fully fledged application for Linux is way more than writing this gigantic documentation. The software works fine for almost all the cities. The oil that runs the software is data. The data is fetched from Foursquare API. The data is free, since we are using regular version of the API. The data that is premium is available at certain subscription rate. We have used simple and lucid technique to plot and use the data for determining the best place according to the preferences given by user. The wisp application software is a light weight and can be used in almost any of the system present out there. The main objective of the Wisp software is to get knowledge about what the place is all about. It may be used by people planning for a tour in an unknown place and give the preferences according to their choice. That may help them to get knowledge in real time about the place and the location of the entire place present in the hot spot.

This software is designed to eliminate the extra overhead of researching about a place according to the preferences of the user and make a real time visualisation. This is a console application; the future scope of this project lies in creating a web based or android base application which shall be used by most of the people.

REFERENCES

1. Pal, Jimut Bahan, Wisp, 2019, Github, Github Repository, https://github.com/Jimut123/wisp, accessed on 29.03.2019.

2. Foursquare Developers, The developer website for Foursquare API, https://developer.foursquare, accessed on 29.03.2019.

3. Folium, The visualisation tool for the Map, https://python-visualization.github.io/folium, accessed on 29.03.2019.

4. SDLC and Preliminary Investigation, Information Technology Blog, http://it-alteration.blogspot.com/2016/02/sdlc-and-preliminary-investigation.html, accessed on 30.03.2019.

5. Preliminary Investigation, Freetutes article on Software Development, https://www.freetutes.com/systemanalysis/sa2-preliminary-investigation.html, accessed on 30.03.2019.

6. The ultimate guide to project planning. Project Manager, https://www.projectmanager.com/project-planning, accessed on 30.03.2019.

7. Project Planning, A blog on Technopedia, https://www.techopedia.com/definition/14005/project-planning, accessed on 30.03.2019.

8. SDLC and Preliminary Investigation, Information Technology Blog, http://it-alteration.blogspot.com/2016/02/sdlc-and-preliminary-investigation.html, accessed on 30.03.2019.

9. How to write a good SRS for your project, GeeksforGeeks, https://www.geeksforgeeks.org/how-to-write-a-good-srs-for-your-project/, accessed on 30.03.2019.

10. Software Requirement Specification Documents with Example, Krazytech, https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database, accessed on 30.03.2019.

11. Advantages of Pert Vs Gantt charts, Lucid Chart, https://www.lucidchart.com/blog/advantages-of-pert-charts-vs-gantt-charts, accessed on 30.03.2019.

12. Gantt chart, Wikipedia article, https://en.wikipedia.org/wiki/Gantt_chart, accessed on 30.03.2019.

13. Gantt chart Vs Pie Chart – What are the Differences?, Edraw article, https://www.edrawsoft.com/difference-gantt-chart-pert-chart.php, accessed on 30.03.2019.

14. What is a Gantt chart? , Gantt.com, https://www.gantt.com/, accessed on 30.03.2019.

15. Folium, The visualisation tool for the Map, https://python-visualization.github.io/folium, accessed on 29.03.2019.

16. What is RAD model- advantages, disadvantages and when to use it, http://tryqa.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/, accessed on 29.03.2019.

17. What is RAD model? Advantages and Disadvantages, https://www.guru99.com/what-is-rad-rapid-software-development-model-advantages-disadvantages.html, accessed on 30.03.2019.

18. Requests: HTTP for humans, Python Software Foundation, https://pypi.org/project/requests, accessed on 31.03.2019.

19. Requests, http://docs.python-requests, accessed on 31.03.2019.

20. Pandas, Python Software Foundation, https://pypi.org/project/pandas/, accessed on 31.03.2019.

21. Folium, Python Software Foundation, https://pypi.org/project/folium, accessed on 31.03.2019.

22. Geopy, Python Software Foundation, https://pypi.org/project/geopy, accessed on 31.03.2019.

23. Numpy, Wikipedia article, https://en.wikipedia.org/wiki/NumPy, accessed on 31.03.2019.

24. Database Design, Wikipedia article, https://en.wikipedia.org/wiki/Database_design, accessed on 31.03.2019.

25. Charrington, Dwayne, Open source database schemas, Github repository, Github, https://github.com/Vheissu/Open-Source-Database-Schemas/blob/master/vheissu-foursquare-schema.md, accessed on 31.03.2019.

26. Breslin, Mike, Foundbite's Data Model: Relational DB Vs. NoSQL on IBM Cloudant, IBM Developer, The developerWorks Blog, https://developer.ibm.com/dwblog/2013/relational-nosql-cloudant-database/, accessed on 31.03.2019.

27. Database Schema, Wikipedia article, https://en.wikipedia.org/wiki/Database_schema, accessed on 31.03.2019.

28. Data Integrity, Wikipedia article, https://en.wikipedia.org/wiki/Data_integrity, accessed on 31.03.2019.

29. Tkinter, Python organisation, https://wiki.python.org/moin/TkInter, accessed on 31.03.2019.

30. Software Engineering Paradigms and models Information Technology Essay, Uni Assignment Centre, https://www.uniassignment.com/essay-samples/information-technology/software-engineering-paradigms-and-models-information-technology-essay.php, accessed on 30.03.2019.

31. Gurendo, Dmitri, Software development Life Cycle, XB software blog, https://xbsoftware.com/blog/software-development-life-cycle-spiral-model/, accessed on 30.03.2019.

32. Michael, C.C., Wyk, Ken, Van, and Radiosevich, Will. (2005), Risk based and functional security testing. Official website for the department of homeland security, https://www.us-cert.gov/bsi/articles/best-practices/security-testing/risk-based-and-functional-security-testing, accessed on 30.03.2019.

33. Test approach. Tutorials Point article. https://www.tutorialspoint.com/software_testing_dictionary/test_approach.htm, accessed on 01.04.2019.

34. Unit Testing, A blog on Software Testing, http://softwaretestingfundamentals.com/unit-testing/, accessed on 01.04.2019.

35. Pal, Jimut Bahan, Github, Github repository,

https://github.com/Jimut123/jimner, accessed on 01.04.2019.

36. Open Street Map of Pune, Served using CDN, available on https://cdn.staticaly.com/gh/Jimut123/prog_backups/8f93bfaf/python/wisp_things/test_scripts/maps/Pune_osm_4.html, accessed on 01.04.2019.

37. Stamen Toner Map of Kolkata, Served using CDN, available on https://cdn.staticaly.com/gh/Jimut123/prog_backups/8f93bfaf/python/wisp_things/test_scripts/maps/kolkata_st_4.html, accessed on 01.04.2019.

38. Mapbox Bright Map of Kolkata, Served using CDN, available on https://cdn.staticaly.com/gh/Jimut123/prog_backups/8f93bfaf/python/wisp_things/test_scripts/maps/kolkata_mb_4.html, accessed on 01.04.2019.

39. Stamen Terrain map of Kolkata, Served using CDN, available on https://cdn.staticaly.com/gh/Jimut123/prog_backups/8f93bfaf/python/wisp_things/test_scripts/maps/kolkata_st_3.html, accessed on 01.04.2019.

40. Mapbox Control Room Map of Canada, Served using CDN, available on https://cdn.staticaly.com/gh/Jimut123/prog_backups/8f93bfaf/python/wisp_things/test_scripts/maps/canada_mcr_5.html, accessed on 01.04.2019.

GLOSSARY