



Group 8:  
Assignment 3 Report

## *Design Document*

Hou, Wenran - 18wh10@queensu.ca #20144514

Huilin Xu - 18hx16@queensu.ca #20151379

Liu, Zhibing - 18zl142@queensu.ca #20170623

Li, Yucan - 18yl259@queensu.ca #20164303

Zheng, Yiming - 19yz38@queensu.ca #20182959

*Course Project*

CISC-498

Oct 30, 2022

# Table of Content (External Design)

## **1. Introduction**

1.1 Purpose

1.2 Content Overview

## **2. User interface design**

2.1 User Flow Diagram

2.2 User View Tour

## **3. Usage scenarios**

## **4. Prototype**

4.1 Prototype Method

4.2 Prototype Design Concept

## **5. Work products**

5.1 Require Input

5.2 Output File

## **6. Integration with existing processes**

## **7. Feedback from customer**

## **8. Requirements update**

# Table of Content (Internal Design)

## **1. Introduction**

1.1 Goal of system

1.2 Overview of structure and contents

## **2. Programming environment**

2.1 IDE

2.2 Programming languages

2.3 Database system

2.4 Source control system

2.5 UML tool

## **3. Software architecture**

3.1 Deployment diagram and UML component

3.2 Architecture Style

3.3 Class based view

3.4 Interfaces of key components

## **4. Data design**

## **5. API design**

## **6. Algorithms**

## **7. Notable trade offs**

## **8. Notable risks**

## **9. Milestones**

# External Design Document

## 1. Introduction

### 1.1 Purpose

In the context of increased Arctic warming, glaciers in the Canadian Arctic have become a significant contributor to global sea level rise. The Ice Climate and Environment Laboratory (ICELab) at Queen's University manages and compiles data from a network of four high-latitude Arctic weather stations that are currently used by Arctic researchers at Queen's University[1]. However, while the lab has over 10 years of data, the data stored in csv format is difficult to navigate and distribute. Imagine you want to find the temperature at 4:30 p.m. on the 28th of last month, it might take you at least 10 minutes to get that one item of data off the top of thousands of rows. In addition to making the data more accessible to internal staff, ICELab also hopes to make this data available to other students and researchers at Queen's University, as well as the larger scientific community, so that their research and projects can use it. Therefore, our goal is to build a web-based real-time data explorer that allows users to register an account and perform a number of actions such as creating dashboards, viewing weather station data, and downloading data. Aimed at non-geographic professional users, our design concept includes a series of simple operations such as visualizing and labeling data with icons. The main function of the site is not only to facilitate the data that ICELab can organize into a more accessible form, but we also hope that the site will address the difficulty of the general public in accessing scientific findings and research data shared by Arctic researchers, and engage the community by making the data better understood and accessible.

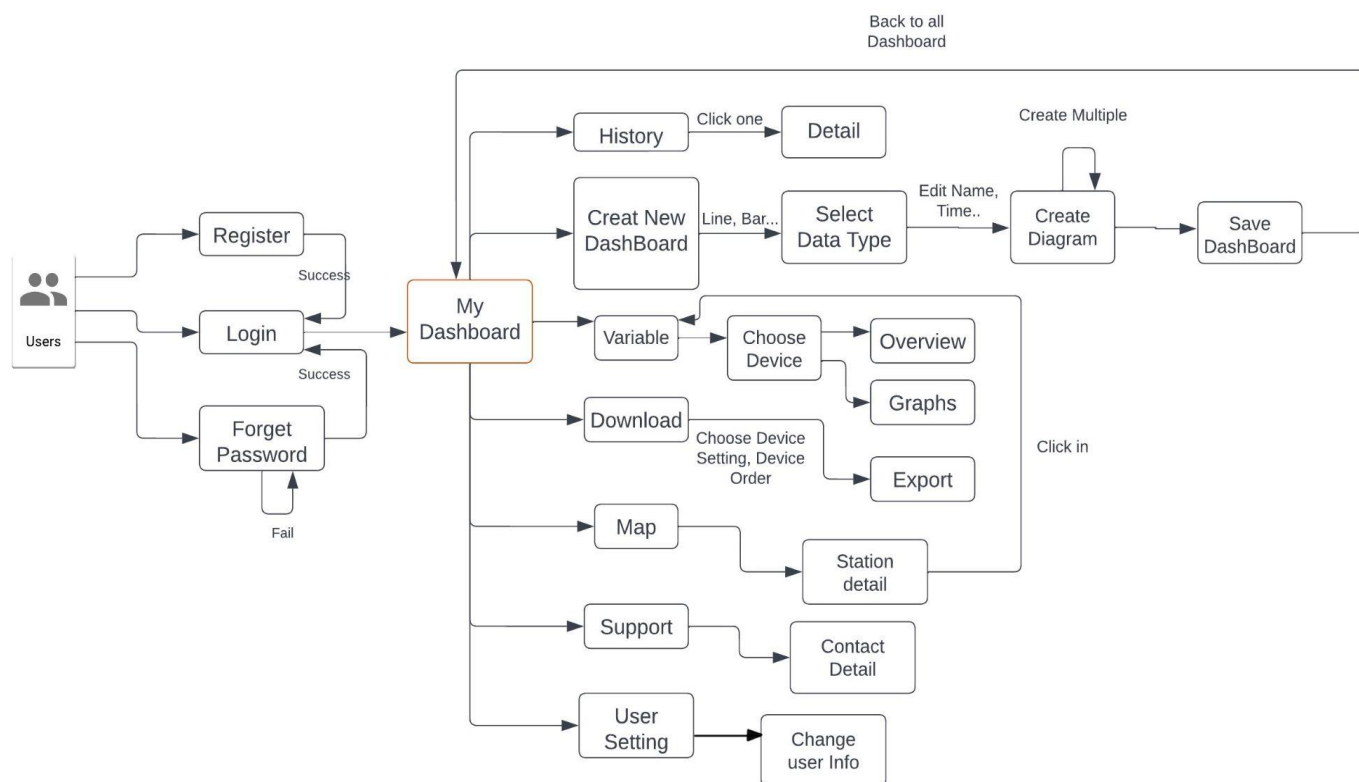
### 1.2 Content Overview

In the External design document, based on customers' requirement, we made a low-fidelity prototype for the explorer. We choose to use a User Flow Diagram as the start of our User interface design, and it is followed with different webpage prototypes in different using scenarios. We also conclude usage scenarios which derived from the user group from the requirement documents. Each single work flow is described in it. After that, we make the design concept and method with our explorer in the Prototype section. In order to find out the and characterize the

boundary of this explorer, work product is proposed by identifying the input and output of this system. During the integration process, we find out our explorer perfectly fits customer's expectations and meets their requirements. We also throw a meeting with our customers to make evaluations with our front-end design at the moment. Better design concepts and features of the system are proposed from the meeting and we update the requirement for the explorer. At the end, we want to solve the UX design philosophy of our project, also including further plans and developments for the subsequent tasks to be performed.

## 2. User interface design

### 2.1 User Flow Diagram



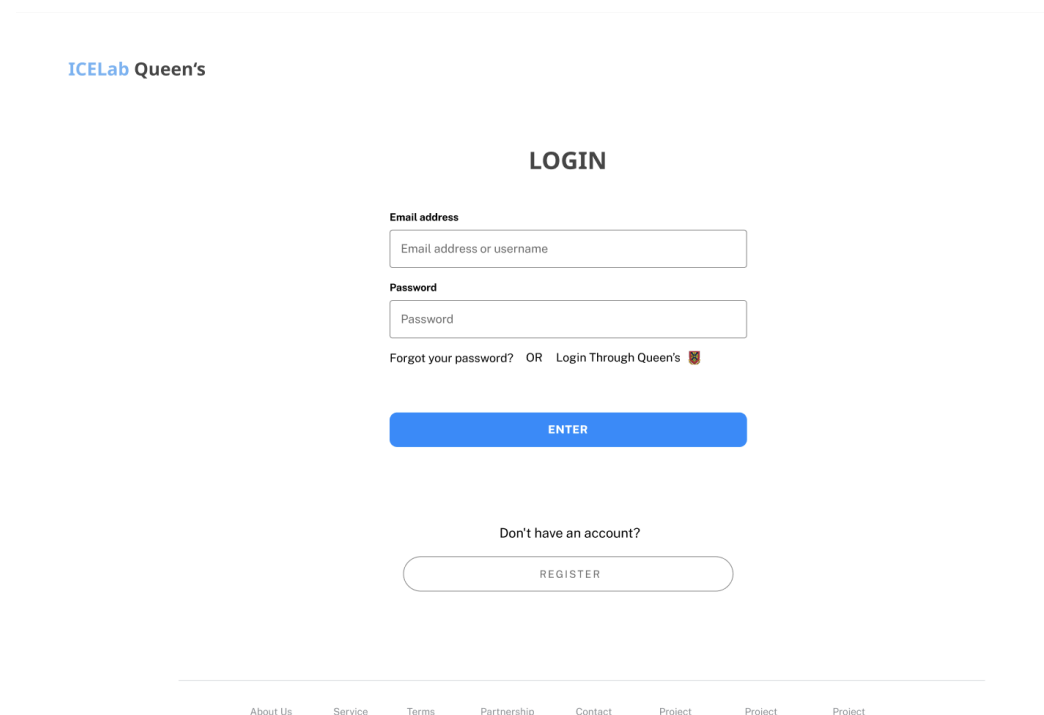
**Figure 2.1 User Flow Diagram**

This is the navigation structure of the site, divided into 9 main sections. Login, Home (My Dashboard), History, Create, Variables, Downloads, Maps, Support and Settings. (Figure 2.1) Each line represents a single action, such as a click or data entry. If it is successful, this would navigate towards the right page. My Dashboard is

the home page of the site and the most important through which I can go to any feature I desire. Below, I'll go into great depth about each phase of the procedure.

## 2.2 User View Tour

When I enter the site for the first time, I will see the login screen with the ICELab logo and I can perform three actions: login, create an account or forget my password. (Figure 2.2) If I choose to create an account, my information will be entered and ICELab will be notified, and after successful creation, I'll be taken back to Login. However, since the site is intended to be more accessible to all, I can also choose to go directly to it without logging in.



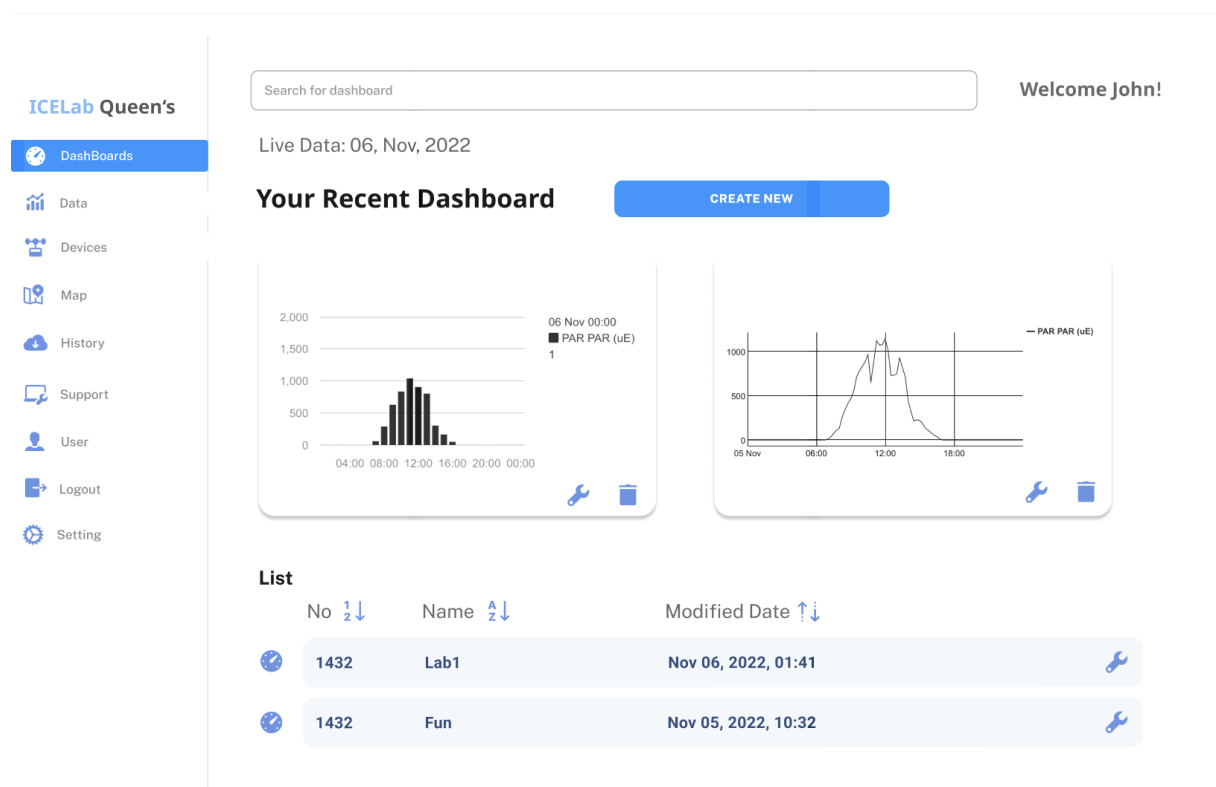
The screenshot displays the login interface for ICELab Queen's. At the top left, the logo "ICELab Queen's" is visible. The main heading "LOGIN" is centered. Below it, there are two input fields: "Email address" with a placeholder "Email address or username" and "Password" with a placeholder "Password". A link "Forgot your password?" is positioned to the left of the text "OR Login Through Queen's" which is accompanied by a small crest icon. A prominent blue "ENTER" button is located below the password field. At the bottom of the login section, the text "Don't have an account?" is followed by a "REGISTER" button. The footer contains a horizontal list of links: "About Us", "Service", "Terms", "Partnership", "Contact", "Project", "Project", and "Project".

**Figure 2.2 Login Page**

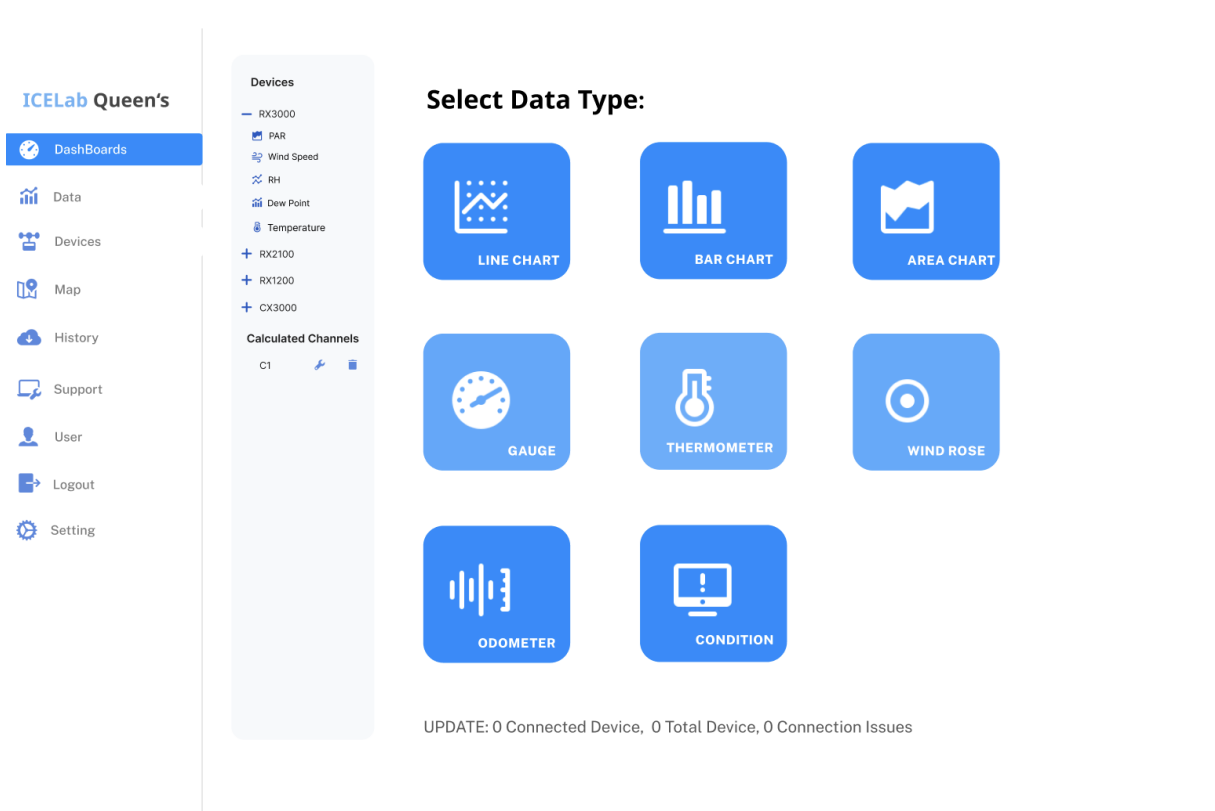
My Dashboard will be the first thing I see after logging in (or not). (Figure 2.3) There will be a line that reads: "You don't have any data yet, please create one" if this is my first time using the website. If I have previously used it and had data to save, I will see the charts from my most recent Dashboard as well as a list, which allows me to

view all of the Dashboards I have created. By selecting the trash can icon on this page, I may easily remove the Dashboard at the same time. There is also a search bar at the top of the main screen where I may look through recorded objects if I can't find the Dashboard I'm searching for. And if I want to jump to exported data on this page, I just need to look to the guide bar on the left side and be directed to all the functions contained within.

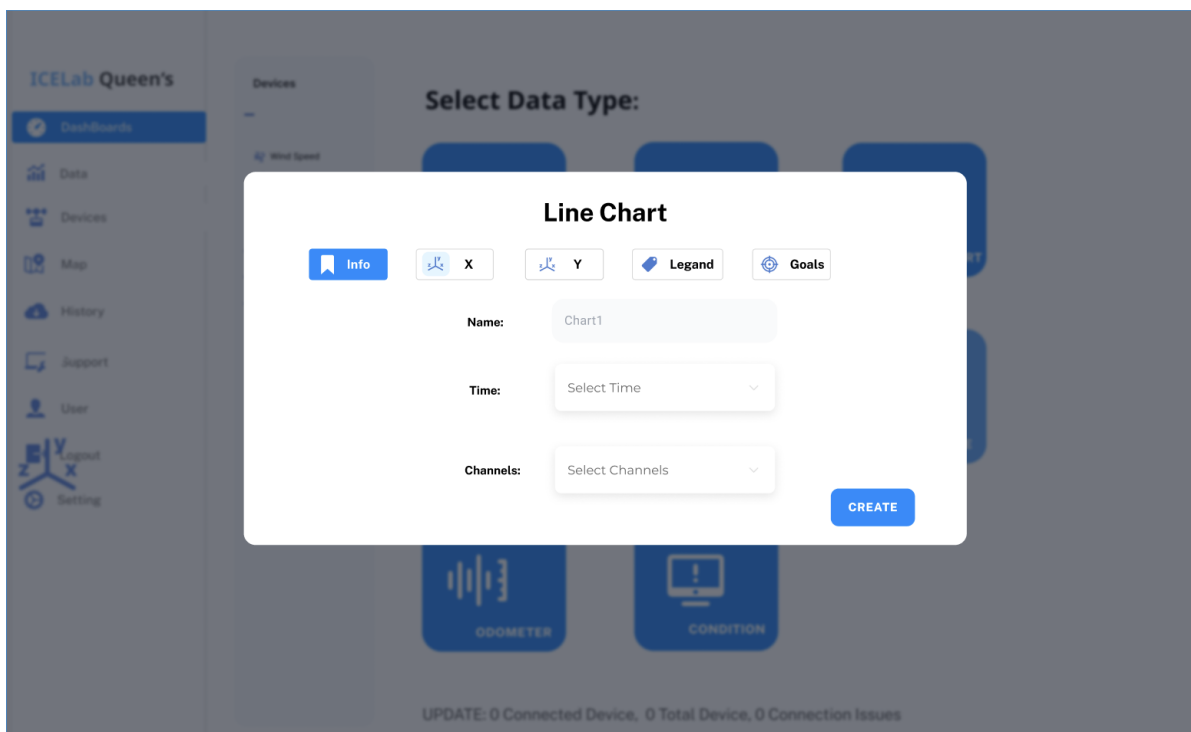
Now I want to create a new Dashboard, so I click on the blue button "Create New" and it will bring me to the new creation page, where I can choose from nine different types of charts. (Figure 2.4) I can select any of them and a new window will pop up where I can add further details like the name of the chart, time zone, etc. Once I click the "Create button" at the bottom, a new chart will be created. (Figure 2.5) I now have the option to keep clicking "Create More" to add more charts or click "Save" to keep the current data and add the Dashboard.



**Figure 2.3 Dashboards Page**



**Figure 2.4 Dashboards Create Page**

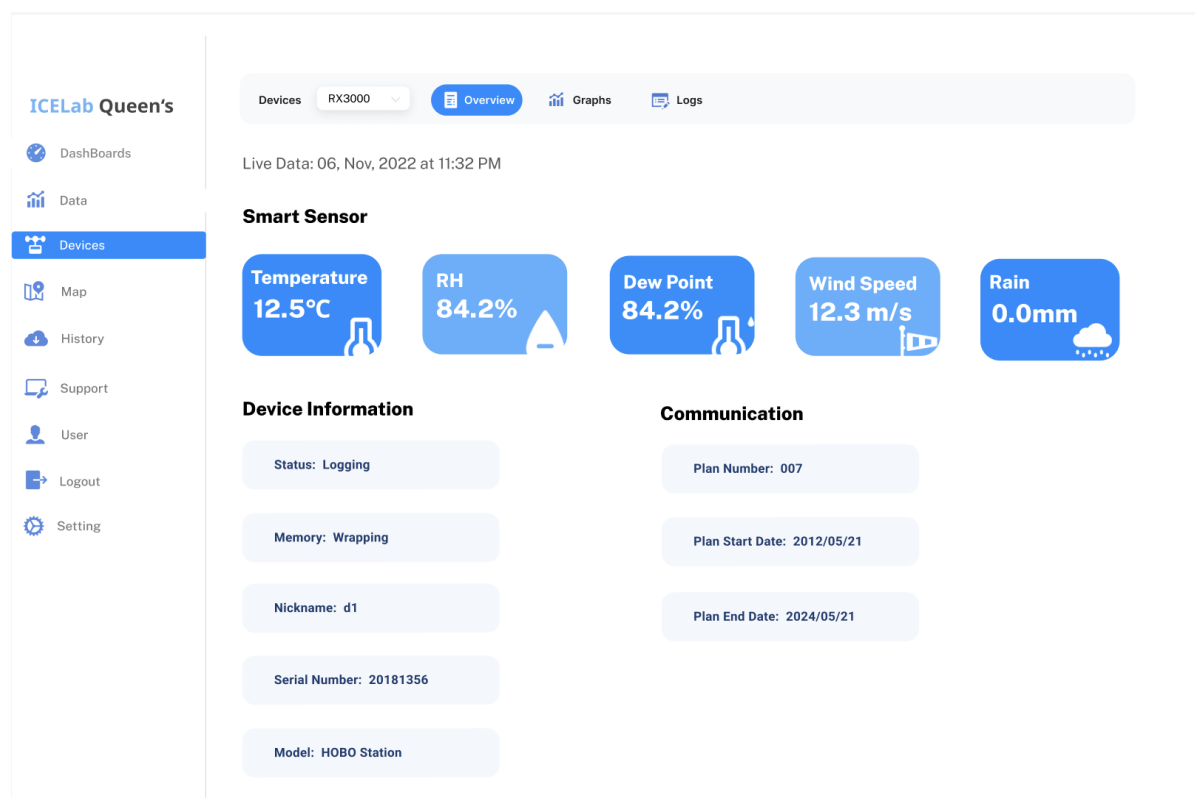


**Figure 2.5 Dashboards Create Page 2**

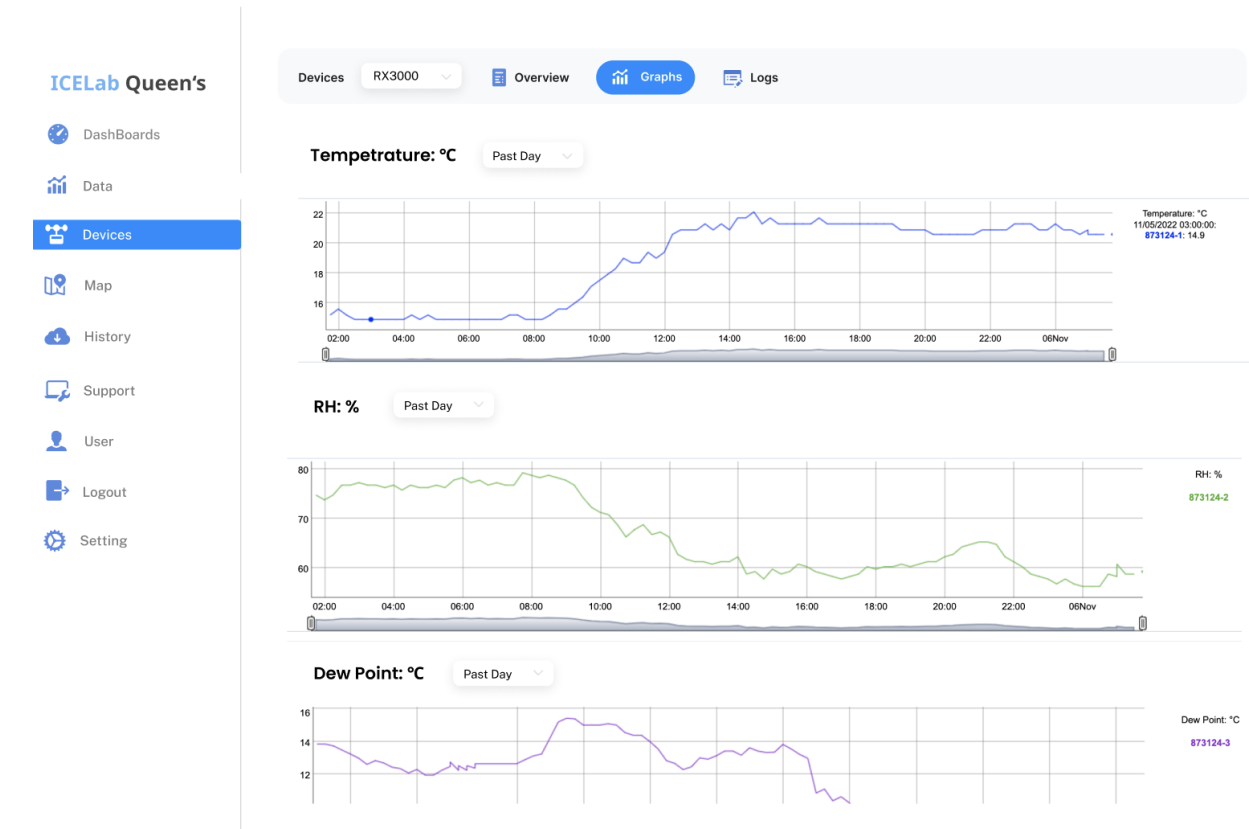


Going back to the left sidebar, if I now want to see the readings for each individual sensor, I can do so by clicking on the device and immediately seeing the summary information. (Figure 2.6) This includes information on the device itself, such as its status, model number, serial number, etc., as well as a variety of values, such as temperature, relative humidity, wind speed, and so forth. According to the gray line labeled "Live Data" on the website, all the data presented on this page will be the most recent data the sensor has ever returned, accurate to the minute.

Through navigating the drop-down menu next to "Overview," I can choose a different sensor if I no longer require the information from this one and instead wish to view Sensor2. There is also a "Graph" button next to "Overview," as seen in Figure 2.7, which will offer a more user-friendly design for visualizing and creating data summaries. I will see various line graphs used to show how the data fluctuates. And next to each type of data, I can select the time period for which I want to display the data, such as a day, a week, or a month.



**Figure 2.6 Devices Overview**



**Figure 2.7 Devices Graphs**

Let's return to any page's left sidebar now, where we may click "Data" to get a page specifically for exporting data. (Figure 2.8) The name of the file to be exported, the file type (such as.csv or pdf), and the duration (for example, I want to export data for a week) are the only three fields I need to fill out. I can specify how many individual sensors I want to export at the bottom of the page, but I can also decide not to since there will be a default value of export sensor 1. Additionally, I may alter the order in which the data is exported. For instance, if I want to export sensor 1 data before sensor 2 data, I only need to drag the blue icon under "Devices Order". Once everything is done, I will click on the blue "Export" button and the file I want will automatically start downloading and saving on my computer.

ICELab Queen's

DashBoards

Data

Devices

Map

History

Support

User

Logout

Setting

Export Setting

EXPORT

Name:

Chart1

File Format:

Setcet Format

Setcet time zone

FROM TO

7 Days

Device Setting

+

RX 3000

RX 3010

Device Order

A Z

↓

RX 3000

RX 3010

**Figure 2.8 Export Data**

Now that I was intrigued about the precise locations of these devices, I clicked on the map in the sidebar to show a map with each device's location prominently indicated on it. (Figure 2.9) Besides that, a white box displaying the most recent three weather data is located above each device. By clicking on any of the devices, I am redirected to the Devices screen for more details.data is located above each device. By clicking on any of the devices, I am redirected to the Devices screen for more details.



**Figure 2.9 Map**

### 3. Usage scenarios

Since usage scenarios should be built based on user group, the list of user group should be identify first and related user scenario is attached:

User Group	Description	Usage Scenario
ICELab Administrator (Professors)	The ICELab Administrator is the group responsible for providing and maintaining the data and regularly reviewing the population of visitors to the site. They are highly skilled and therefore only need to provide help in using the site initially. There are no additional packages to install for this.	Dr. Thomson and Dr. Omelon are professors at Queen's University. (1) They used their assigned administrator account login to the explorer. (2) Professors first check if there is anyone request to open an user account for browsing data over the explorer and grant them permission. (3) After that, they upload the latest research data collected from the Lab report. (4) Through tracking one user's activity, professors find he is suspicious for requesting to download data hundreds of times, so a warning

		email is sent to the user and the account is frozen for hours. (5) At the end of the day, professors check the survey to see if there are any suggestions or recommendations that users point out.
Queen's University Students	Queen's students are those who wish to download Arctic data for use in completing coursework, and they need to use their own queens account to log in. However, they are very computer literate so they do not need additional help to access the site	Jack is a Queen's University student who is interested in Arctic weather changes. He uses his Queen's account to get authorized login into the explorer. He finds an Arctic research station and tries to download weather data within this month. As soon as he gets a csv file he leaves without logging out his account, but the explorer automatically logs him out after he becomes inactive for 20 minutes.
Arctic Environmental Researchers	Arctic Environmental Researchers is a group of people who want to use Antarctic data for scientific purposes. They can select the time period for data and generate graphs from the web page, as well as view the latest data. They are also very familiar with the technology and do not need additional guidance.	Crystal and Mike used to be Arctic Environmental Enthusiast and became Arctic Environmental Researchers recently. Today they would like to collect some data from explorers to make some academic analysis. So when they log in their permitted account. They first select their interest period from 2000 to 2010 and look for the average rainfall during the period. They used the built-in calculation tool to find out the average and download the analyzed result in pdf format.
Arctic Environment	Arctic Environment Enthusiast is a group of people who want to use data for their personal interests, such as generating their own weather logs. They can also download and search the data at any time, but need permission from the administrator. They do not	

Enthusiast	need to download additional packages and are considered to be skilled computer users.	
------------	---	--

## 4. Prototype

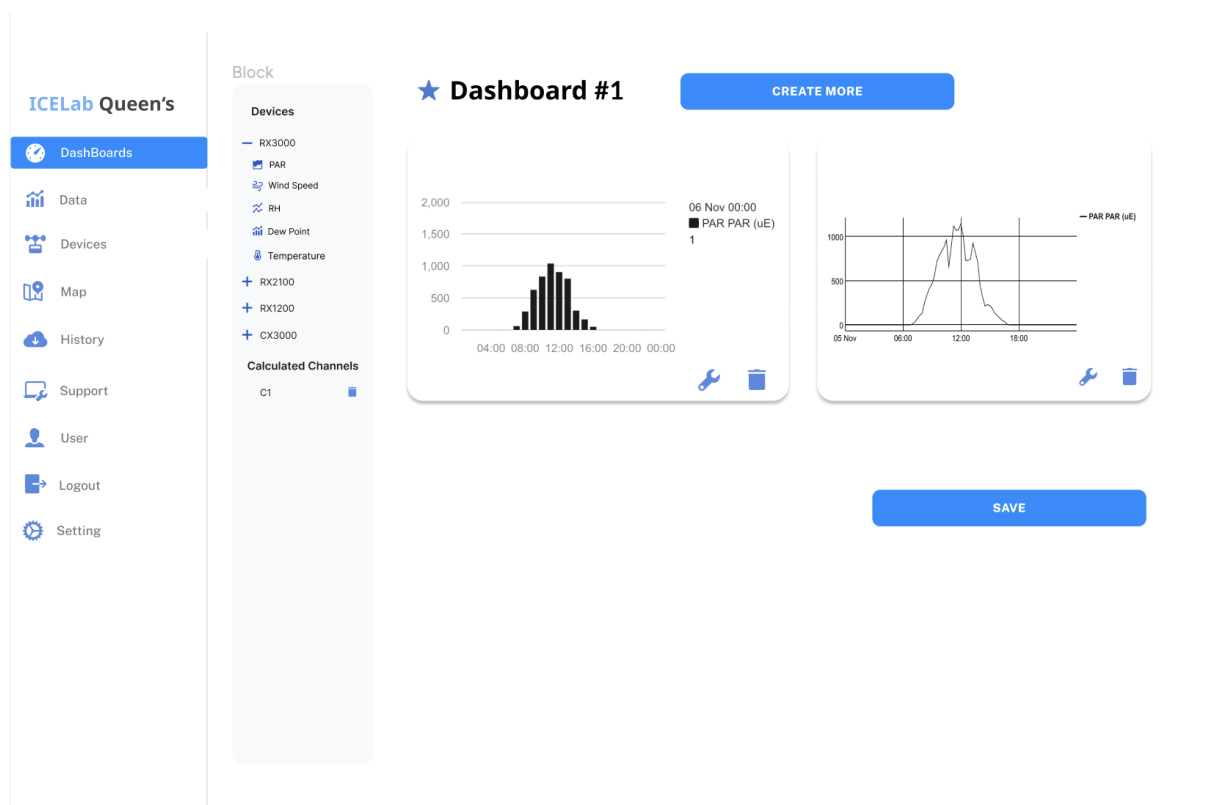
### 4.1 Prototype Method

Our prototype was created on the basis of the application figma, which contains all the features and online collaboration support we need. All we have to do is visit the figma website and create a new file to start implementing our idea. Since the platform we chose is a web page, this is also the platform we will use for testing. In the top right corner of the diagram, you can see a small play button for the title. Once you tap it in, you can interact like a normal website and try out all our features. You can also go directly to this URL(  
<https://www.figma.com/file/h6OqYGj28BYTM2SncepR3t/CISC498?node-id=0%3A1&t=VQ4I3MJ1POxLMWsj-0>) and go directly to the test screen. Unfortunately, figma does not implement all the features and has some known bugs. Our prototype has many buttons that "appear" to be clickable, but due to the frame limit, we don't actually have the means to do so. Also, we can't really let the user create an account, because it also depends on the backend database and we can't implement such a complex feature with figma.

### 4.2 Prototype Design Concept

The GUI prototype design of this web application adheres to general GUI design guidelines in terms of optimized operation, minimal memory, pattern avoidance and consistency. The operation process is straightforward, quick and error-free, and the user only needs to click on "Add" and the ability to add an item in most cases. The user just has two executable buttons, as seen in image 4.1 below, allowing them to quickly access the most important information. For these operations, there are no complex commands, which means that everything is simple and neat so that users can quickly learn how to use it. The layout is also consistent in terms of color, formatting, and navigation commands. Beyond the design, it was critical to consider how to make each button clearly visible and guided. For example, most professional

terms are accompanied by illustrated icons to make it easier for users unfamiliar with these terms to quickly become familiar and get started. Designing an efficient interface was also key; we minimized the number of actions available to the user and attempted to accomplish as many functions as we could with as few buttons as feasible. In addition, the team monitored the flow of the interface, such that clicking a button led to the correct screen, rather than something that was not expected. To avoid undesirable features, the team had to regularly calibrate each feature, even if only minor changes were made.



**Figure 4.1 Sample Web Shown**

## 5. Work Products

### 5.1 Require Input

The explorer itself is simply a data presentation and statistical analysis system as far as any work product needed to run the system is concerned. When the web pages are first run, we will put existing data into a database, for example importing the last 5 years of data, which users can access at will. However, since the data will be updated every hour, almost every study dataset will have to be set up by ICELab

personnel to be uploaded automatically or updated manually at a later stage. The study data provided by ICELab are located on local computers in Dropbox and Queen, and these files store Arctic study data in .csv format, containing air conditions, wind speed, and peak gusts, among other statistics. A one-year dataset requires approximately 5,000 kilobytes, and there are up to 20 years of such study data. Therefore, an administrator page will be designed later to facilitate the uploading and management of data by ICELab, and a dedicated account will be required to do so. However, all files should be uploaded by ICELab's internal staff, and none of the features of the entire explorer site require files uploaded as a user.

## **5.2 Output File**

As for the work products that can be generated by the system, they can be operated by both administrators and regular users. For those users interested in Arctic weather conditions, they can select their preferred research datasets for a particular period and choose to download them in csv format. We hope to support more functional download formats in future developments, such as pdfs containing pictures of data trends, not just raw data. When the user selects Export, these downloads will be automatically stored to the PC. Or they can simply take a screenshot on the graph they generated

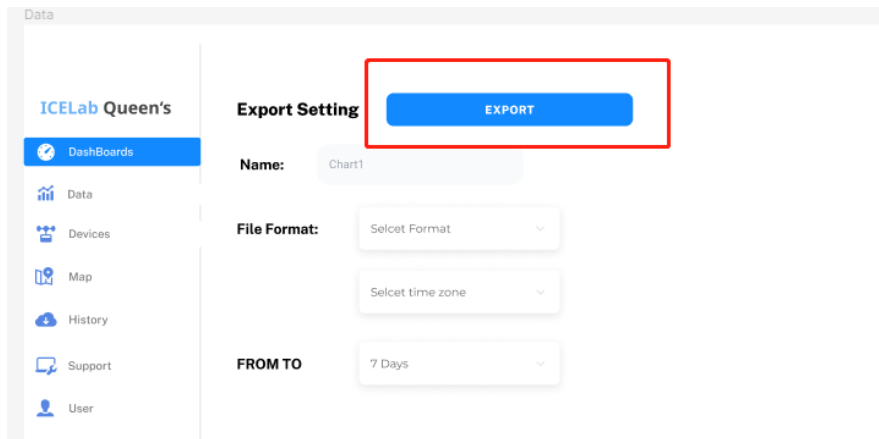
## **6. Integration with existing processes**

The aim of this project is to assist our customers find an alternative of manually sending researching data to people interested in the community. When our customers received a one-year Arctic researching data request, they used to chop them off from a 20-year researching data raw file and select the specific interest field by hand, which lacked efficiency and was time consuming. Besides, our customer collects data from a website called HOBOLink as well. This site is not an open source site as well and if people are interested our customers have to manually grab data as well. So we integrate several features from the site and add new features to the data explorer to make everyone see and make their own analysis on this open public website. Our customers used to collect data from a dropbox account and share research data with people needed. Now they just upload the data to the explorer and keep everything up-to-date.

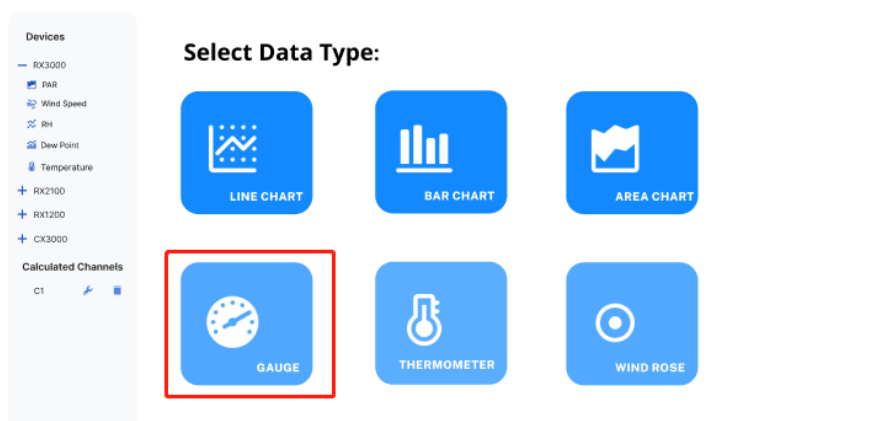


## 7. Feedback from customer

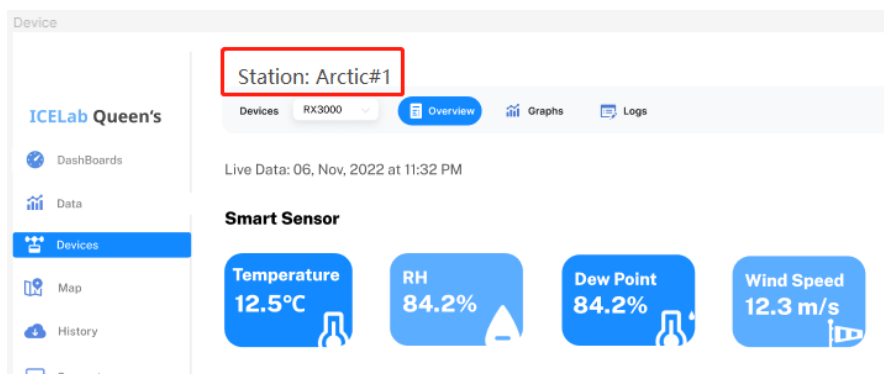
1. Export Setting Page: customers expect the Export button to be placed at a lower level of the page to match the user's intuition and to make operation easier.



2. Select Data Type: Users wanted to remove "Gauge" as a data type because it was not a function of the data they were measuring. In order to assure accuracy, more communication with the customer will be required in the future for the specific data this section contains.



3. Add “Station” category behind each “Device” (a station have several devices)



Since there numerous stations involved in the collecting researching data and each of them has multiple devices or sensors used in measuring. So it should be better to specify that the data is collected from which device and station. (For example, Station1 may have 3 sensors)

4. Public users should be able to access the Dashboard, but only registered, authorized users should be able to download the data.

According to the customer, users are only prompted for permission while downloading data from the explorer. Therefore, logging in does not need to be done in order to just browse the overview of Arctic station sites and associated weather statistics and charts.

5. Making the “Map” first page of the explorer

The customer requested that a map page rather than a Dashboard, which displays the Arctic stations and some basic weather information about them, be the first page the user sees when accessing the website. The map will also indicate the general location of these stations.

6. Adding administration page

To handle data and track web page traffic, ICELab would like to provide a separate administrator account view. In addition to being able to check how many users have recently visited the page and accept new user registrations, it will also be simpler to upload new data and manage existing data. It is expected that ICELab researchers will only need to enter the correct

administrator account and password to log in to the administrator account; after doing so, they will be taken to the administration page.

These will be initial changes that the team needs to make the interface more user-friendly and interactive based on the original purpose, with some added details and tweaks to suit the user's intuition. The overall flow of the interface will also be expected to change based on further customer needs.

## **8. Requirements update**

The first page of the explorer would be the “Map” page instead of a “Login” page, because everyone on the internet can view the basic information and research data through the explorer as long as they are not downloading raw data from the website. Besides, the webpage will add the new feature which mandatorily asks users to register or login into their own account in case they are interested with the raw data on our webpage.

# Internal Design Document

## 1. Introduction

### 1.1 Goal of system

Our goal is to create an efficient, user-friendly weather data system that is easy for the public to explore. High efficiency means that our website should respond quickly to the user commands and give corresponding results. For example, visiting the website, switching pages, downloading data, generating images, and creating an account. Users should not spend too much time waiting for responses other than researching the data. User-friendly means most users should not have difficulties learning the fundamental website function, such as viewing and converting data into charts.

### 1.2 Overview of structure and contents

Our whole system uses layered architecture, which has three tiers in total, namely surface tier, logical tier and data tier. Our objects are the user, Gui, arctic data, and data processor. The entire internal design document contains a detailed introduction of the system architecture, development environment, data type, algorithm usage and notable tradeoff.

## 2. Programming environment

### 2.1 IDE

We are aiming at using Visual studio code and IntelliJ, Two types of software that are popular and powerful.

### 2.2 Programming languages

For front-end design, we need to use mainstream languages such as HTML, CSS, and Javascript. These languages have a long history of development and are still used by many companies in the market. They have great advantages in terms of querying relevant information and completion of work. For the back-end design, we plan to use python. This is because python has unique advantages in data

processing and drawing, including complete external resource packages and easy to understand logic. We also need knowledge related to databases to build up one for providing service to our explorer. So some basic SQL and JavaScript database related knowledge should be known as well.

## **2.3 Database system**

For the database, we aim to use the MongoDB database system. MongoDB is a document database used to build highly available and scalable internet applications. With its flexible schema approach, it's popular with development teams using agile methodologies.

## **2.4 Source control system**

The entire project code and development will be hosted on Github. It's the most useful and popular platform in the world right now. Each of us will clone the code locally through the Git toolkit to make changes, and merge everyone's code into the overall project through a Pull request. Each request needs to be reviewed by two people to ensure that the code merge does not cause problems for the original system.

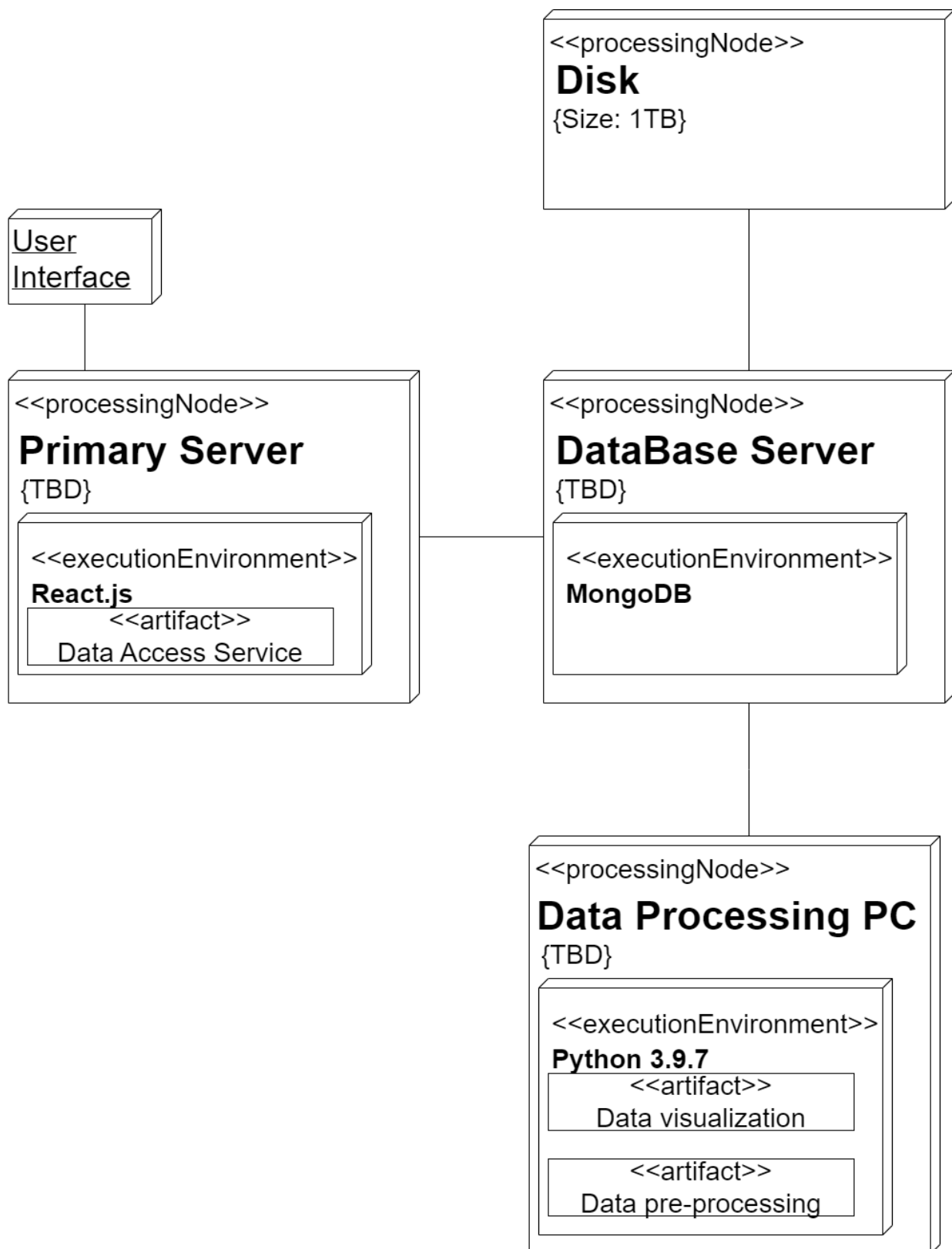
## **2.5 UML tool**

For the graphics tool, we use draw.io, an open source, mature product that allows us to export the finished images in different formats. And the product itself has a lot of UML image templates, which can help us draw appropriate images.

# **3. Software architecture**

## **3.1 Deployment diagram and UML component**

In the development view, we assume that we will need 4 primary pieces of hardware: the primary server which holds the operation of the website which is powered by an infrastructure of react.js and provide data access service in GUIs. the Data Server stores and manages the data in the environment of MongoDB, a 1TB disk is attached to it. The Data Processing PC processes the data and visualizes them into .jpg or other formats that are expected by react.js.



### 3.2 Architecture Style

For the architecture Style, we choose to use layered architecture because in our system the services can be organized hierarchically. As described in the graph each layer provides service to the layers above and depends on the layers below.

Our layered architecture style contains 3 Tiers: the 1<sup>st</sup> tier is the top-level of the application which provides the user interface. This tier translates the task and result into formats that the user can easily understand. The 2<sup>nd</sup> layer is the logic tier, it makes the logic decisions and evaluations which moves the process data between the 1<sup>st</sup> Tier and 3<sup>rd</sup> Tier. In our style, the 2<sup>nd</sup> Tier does the jobs of presentation, application and persistence. (the presentation is placed in the logical tier in our system because the input control section within it will have some calculation in our Data explorer). The 3<sup>rd</sup> Tier is the data tier which stores all the data used in the system.

For the 1<sup>st</sup> Tier: the component is the User Interface which contains the GUI for both user and administer, and the APIs gateways. In the 2<sup>nd</sup> logic Tier it provides a presentation which controls the input and visualizes the data and does the calculation for the data. The application provides the service of presentation application and persistence. The presentation controls the input, visualizes the data; the application component processes the data by the requirement given by the inputs; the persistence component saves the data in entities. In the 3<sup>rd</sup> Data Tier it stores the data that is used in the above. This tier also provides service of controlling the 3<sup>rd</sup> party APIs(from other weather stations).

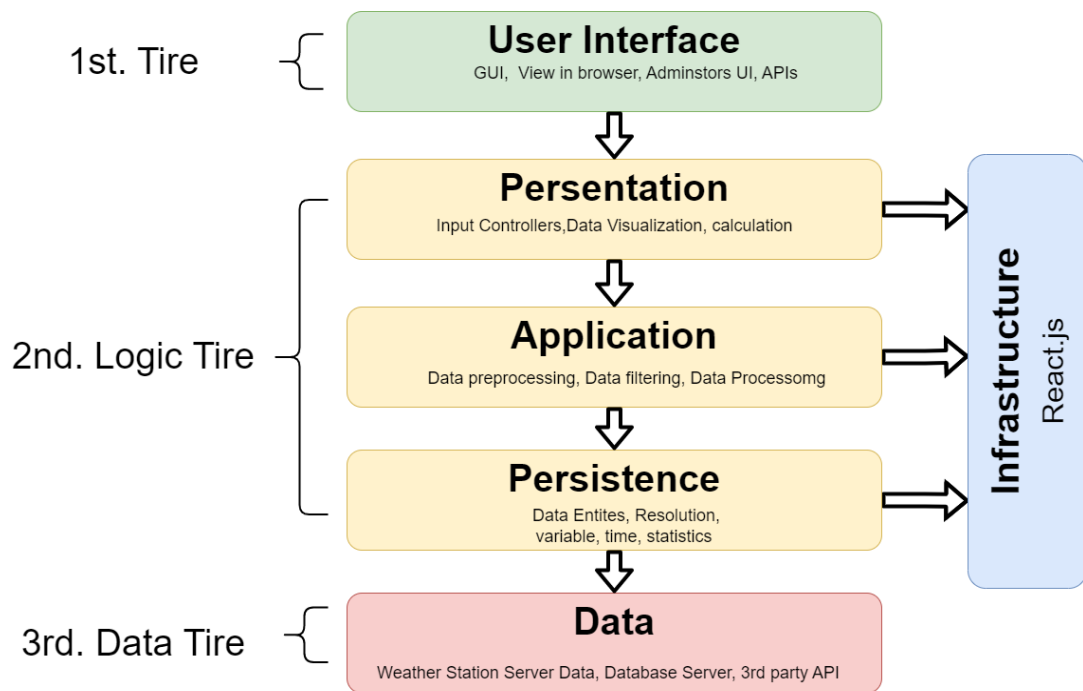


Figure: Explorer Architecture Diagram

### 3.3 Class based view

A class diagram is a UML diagram type that describes a system by visualizing the different types of objects in it and the static relationships that exist between them. It also describes the operations and properties of the class. There are three main kinds of relationships, namely Inheritance, Composition, and Aggregation. Inheritance refers to one class (child class) inheriting the identical functionality of another class (super class) and then adding new functionality of its own. Composition models the part-whole relationship, every part may belong to only one whole, and if the whole is deleted, so are the parts. Aggregation expresses a relationship among instances of related classes. It is a specific kind of Container-Container relationship.

Our class diagram has a total of four objects. User, GUI, data, and data visualizer. Among them, registered users, website visitors, and administrators inherit all the parameters and operations of users, but they also have their unique properties. For example, only registered users can download data, website visitors need to provide their institution, and administrators can track users' download records and operate on data. GUI is a graphics-based operating system interface that uses icons, menus,



and a mouse to interact. In our system, the administrator has its own unique GUI with more functionality and can manage user accounts.

The data processor and filter work together. The filter is responsible for sending the range and conditions defined by the user to the processor, and then the processor calculates the data result or generates the image according to the user's requirements to the GUI, and finally presents it to the user's screen.

Data is a Whole object, and there are four kinds of data that inherit it, namely device, weather, radiation and calculated data. Device data refers to the type of computer, ID, etc. that the user uses to log in to the website. Radiation and weather data are specific values detected and returned by the sensor of the Arctic weather stations. Calculated data is the manipulation and calculation of data over a period of time.

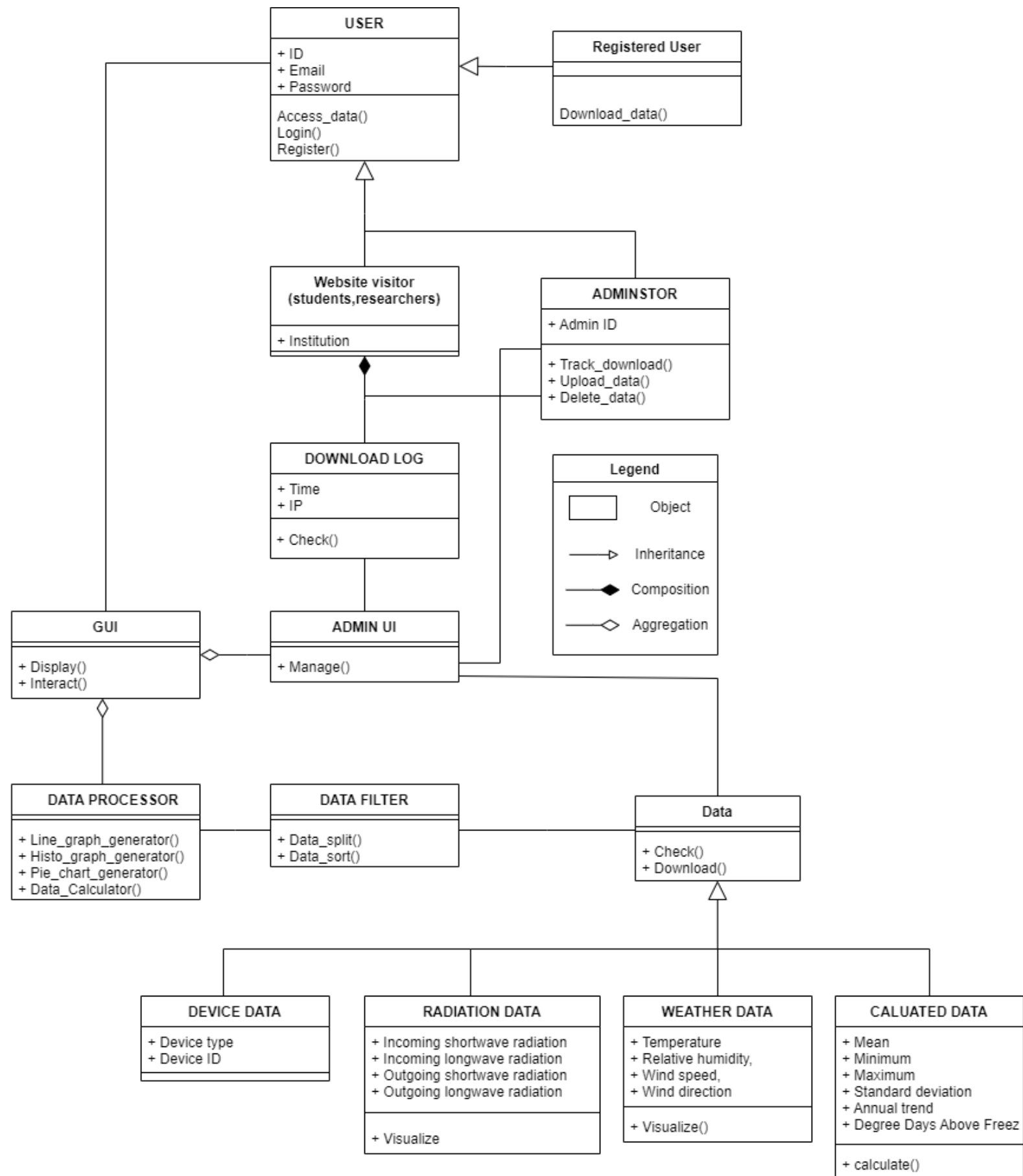


Figure: Class Diagram for each object involved in systema

## 3.4 Interfaces of key components

### 3.4.1 DataBase:

- **Input:**

INT-1: weather data

INT-2: radiation data

INT-3: account data

INT-4: download logs

- **Output:**

OUT-1: data location

- **Pre-condition**

PRE-1: Enough space for input in the disk.

**Ppost-condition**

POST-1: Data is stored in the given location.

- **Exception**

Exp-1: Incorrect data type

Exp-2: Manual Interrupt

Exp-3: Store overtime

### 3.4.2 Data Filter:

- **Input:**

INT-1: Data location

INT-2: Time Range

- **Output:**

OUT-1: Splited Data

- **Pre-condition**

PRE-1: Data in the location given is valid

PRE-2: Inputed Time Range is vaild

- **Post-condition**

POST-1: The output Splitted Data fits the input requirements

POST-2: The data in the database remains the same

- **Exception**

EXP-1: Data missing.

### **3.4.3 Data visualization:**

- **Input:**

INT-1: Data

- **Output:**

OUT-1: Different kinds of Graphs

- **Pre-condition**

PRE-1: Input time range is valid

- **Post-condition**

POST-1: Users successfully receive the graph they request.

- **Exception**

EXP-1: Website crashed

EXP-2: Data missing

### **3.4.4 GUI:**

- **Input:**

INT-1: User interactions

INT-2: Administer command

- **Output:**

OUT-1: requested data

OUT-2: result of commands

- **Pre-condition**

PRE-1: Administer's ID has been verified for administer command

PRE-2: Website server is running

PRE-3: User's Internet connection is stable

- **Post-condition**

POST-1: Users successfully receive the service they request.

- **Exception**

Exp-1: Server under maintenance

## **4. Data design**

For the data in this project, we prefer to use a combination of input and output files and databases for data interaction. Researchers cannot only upload data from existing monitoring stations to the website, but they can also download climate data for specific periods. Because of the high level of internal detail in the data provided by the climate monitoring stations, the website stores the uploaded data in a database.

The massive amount of data needs to be organized uniformly. Therefore, we decided to solve this problem using BNF, the Backus–Naur Form, a context-free language widely used in programming languages, instruction sets, and syntactic representations of communication protocols. Besides, this rigorous, concise, and easy-to-read model of language structure described by formal rules are well suited to show the logical relationships between individual data. The following is the BNF we have built for this project.

<station>::=<location><time><resolution><variables><statistics>

<location>::=region | latitudeLongitude

<time>::=startEnd

<resolution>::=original | <period> | entireDataset

<period>::=monthly | hourly | daily | weekly | annually | seasonally

<variables>::=temperature | relativeHumidity | windSpeedDirection | <radiation> | distance

<radiation>::=<outgoingRadiation> | <incomingRadiation>

<outgoingRadiation>::=outShortwave | outLongwave

<incomingRadiation>::=inShortwave | inLongwave

<statistics>::=windFrequency | mean | standardDeviation | <extreme> | annualTrend | degreeDaysAboveFreezing

<extreme>::=minimum | maximum

In this form, <> refers to a non-terminal (or variable). ::= means that the non-terminal on the left will be defined as an expression on the right. Another symbol in this is |, which indicates selection. [2] We also draw a syntax diagram about this form to make it easier to understand.

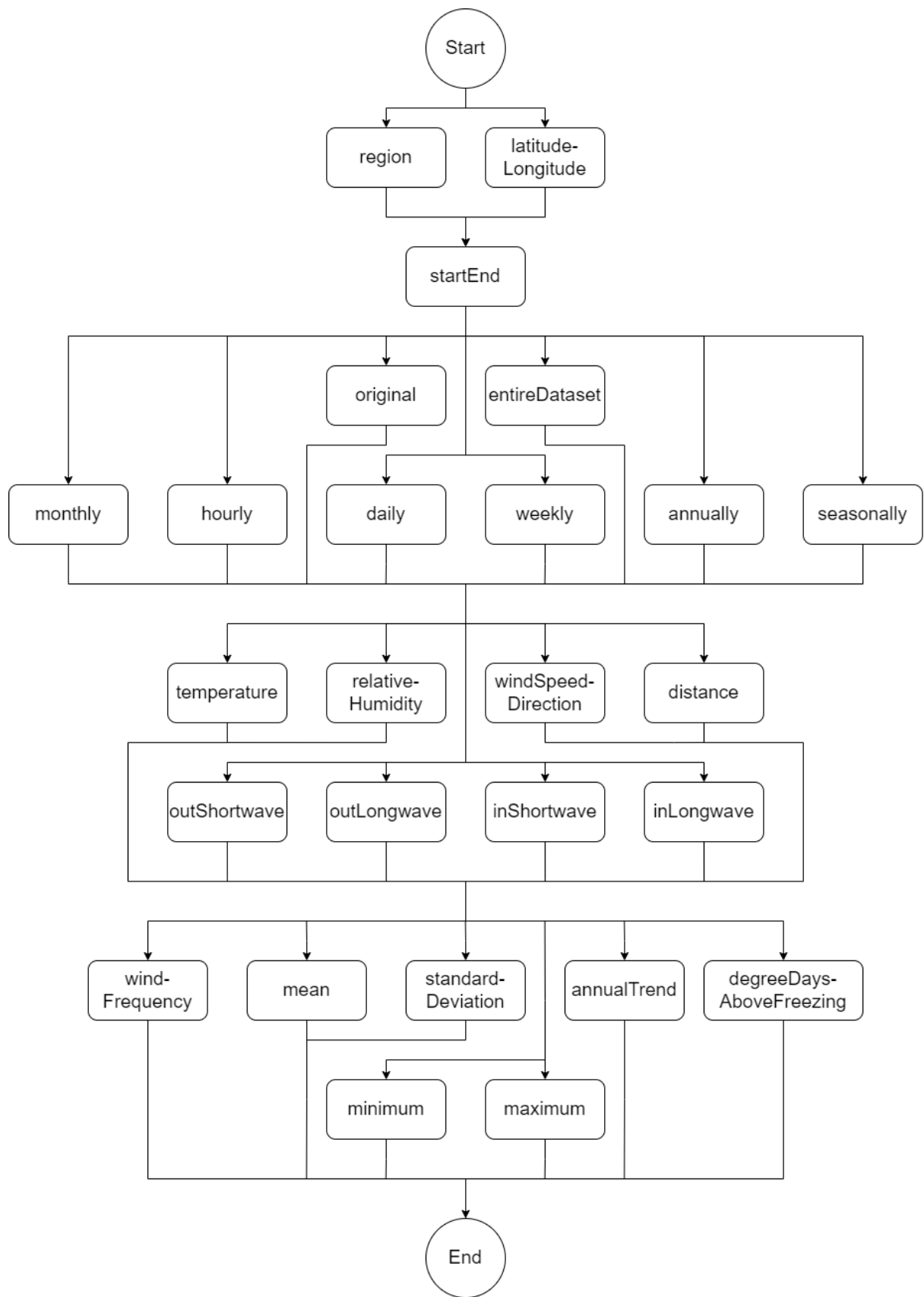


Figure: Syntax diagram of BNF

According to the above BNF and syntax diagram, the station comprises five non-terminals: Location, Time, Resolution, Variables, and Statistics. Location includes two terminals: region and latitudeLongitude. The start and end times determine the content of the non-terminal Time. Two terminals (original and entireDataset) and a non-terminal period containing six terminals make up the Resolution. Variables can be replaced by four terminals (temperature, relativeHumidity, windSpeedDirection, and distance) and one non-terminal radiation. Statistics contains five independent terminals and one extreme non-terminal. After choosing these non-terminals several times, we can obtain a "sentence" consisting of 5 terminals. Thus, this method allows us to precisely get the data we want.

We have chosen a CSV format instead of an XML format for the data format. There are three reasons for this. First, CSV (comma-separated values) is a plain text file format for storing tabular data. Therefore this will reduce the learning cost for the researcher. Second, the content of the data held in CSV is more compact than in XML. CSV files will take up less storage space for the same data. It is suitable for solving the more detailed data in our project. Finally, from talking with customers, we learned that the data that researchers get from the station is in CSV format, so using CSV format is more conducive to interacting with researchers.

In order to describe the concept of real-world relationships, ER diagrams are more appropriate. In our ER diagram, there are five main entities included in it. Location provides the data collected by the monitoring station for Variable, and Time limits the scope of the attributes in Resolution and Variable. At the same time, the Resolution decided by the user will change the content of the Variable under the influence of Time. In addition, by combining the values of Time and Variable, the computer can calculate the values of the different attributes in Statistics.



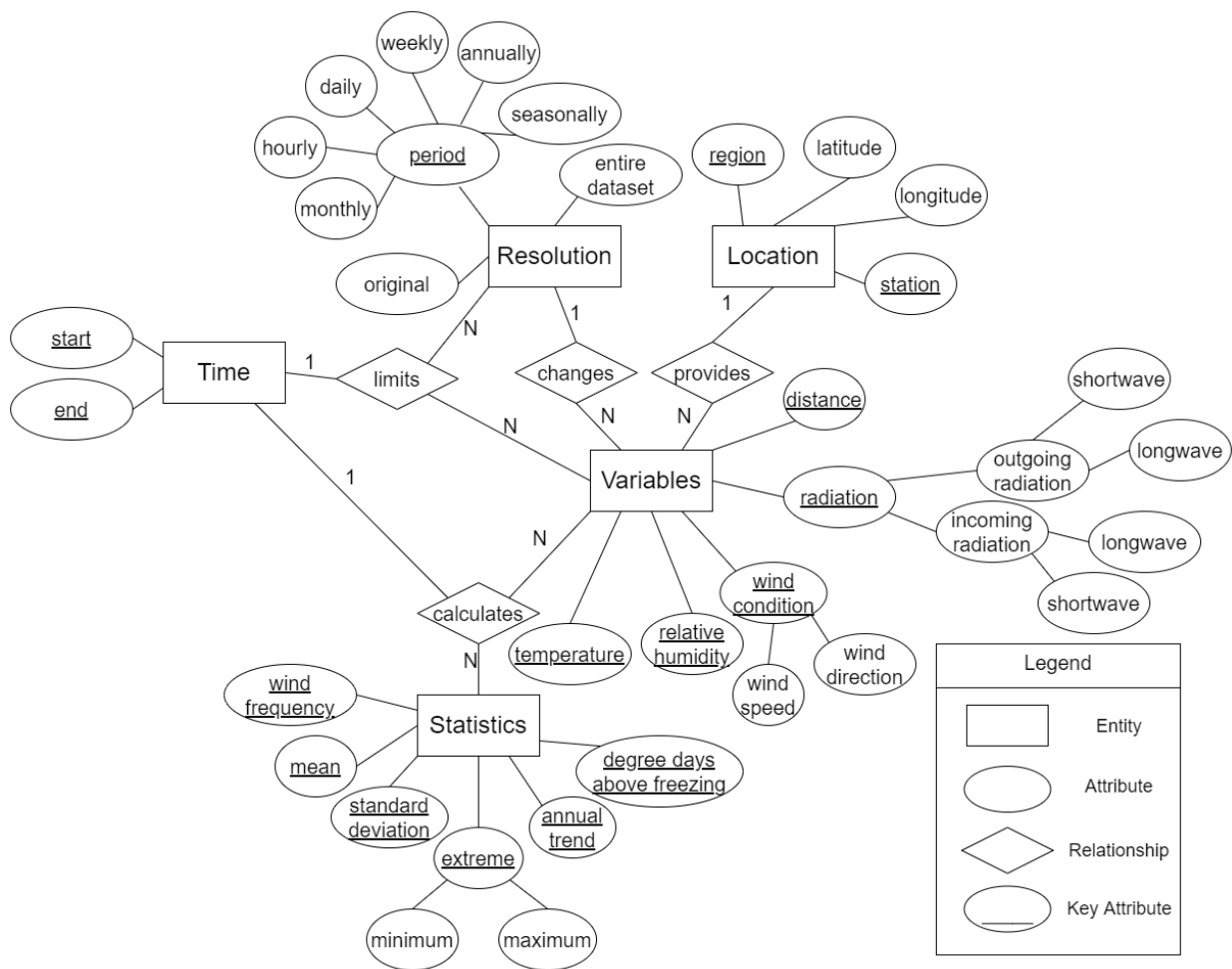


Figure: ER Diagram of Web-based Real-time Data Explorer.

We build the relational schema by analyzing the entities and relationships in the ER diagram above. The source of the data is needed first. As we can see from the figure, Provides serves as a relationship that provides the climate data from the Location entity for the Region and Station attributes to the Distance, Radiation, Wind Condition, Relative Humidity, and Temperature attributes in Variable, respectively. After this, we need to restrict the data in terms of time. The Time entity determines the Period property in the Resolution entity and the properties in the Variable other than Distance by the start and end times. In addition, the Resolution entity also changes the range of each property in the Variable through the Period property. To present the data to the user in a better way, we also need to compute the data. We can calculate Wind Frequency, Mean, Standard Deviation, Extreme, Annual Trend, and Degree Days above Freezing based on the Time and Variable entities.

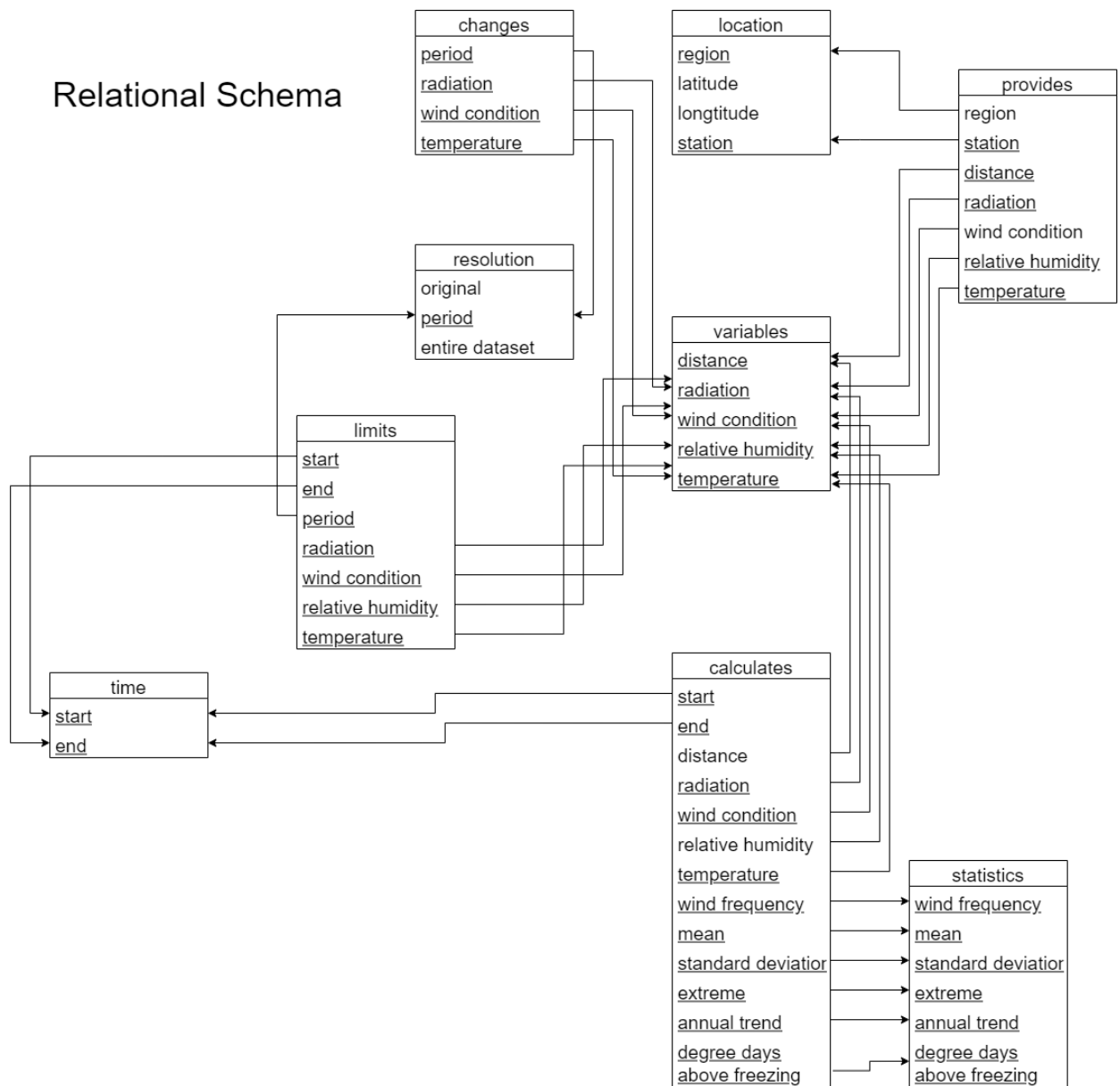


Figure: Relational Schema of Web-based Real-time Data Explorer.

## 5. API design

The customers thought the API would be a cherry on the cake, it is fine if we are not providing this feature at this moment. So if we have more time after we finish all the system, then we will discuss and promote the API to the public. Computing experts would be glad to see if relevant API provided.

## 6. Algorithms

Since this project is to compute and map climate data, the algorithms mentioned next are non-standard. The data processing is divided into three main categories: first, simple data calculation, such as finding the mean, standard deviation, and extreme values. The second is the analysis of the data and the plotting of line graphs. The third is to plot the polar diagram from the wind direction data. Based on Python (the main reason for choosing Python is that we think Python is better and better at processing data and plotting images.), the first case can be solved by calling some common functions, so we will not go into details here. The second case is slightly complicated. Here is an example of temperature over time data analysis. First, we need to read the monitoring station's temperature and date data from Variable and convert the data to DateTime format. Immediately after that, we create a canvas and a subplot. Fill the subplot with data, where the x-axis is the time, and the y-axis is the temperature. After setting the colors, finally use the show method to view the graph (shown below).

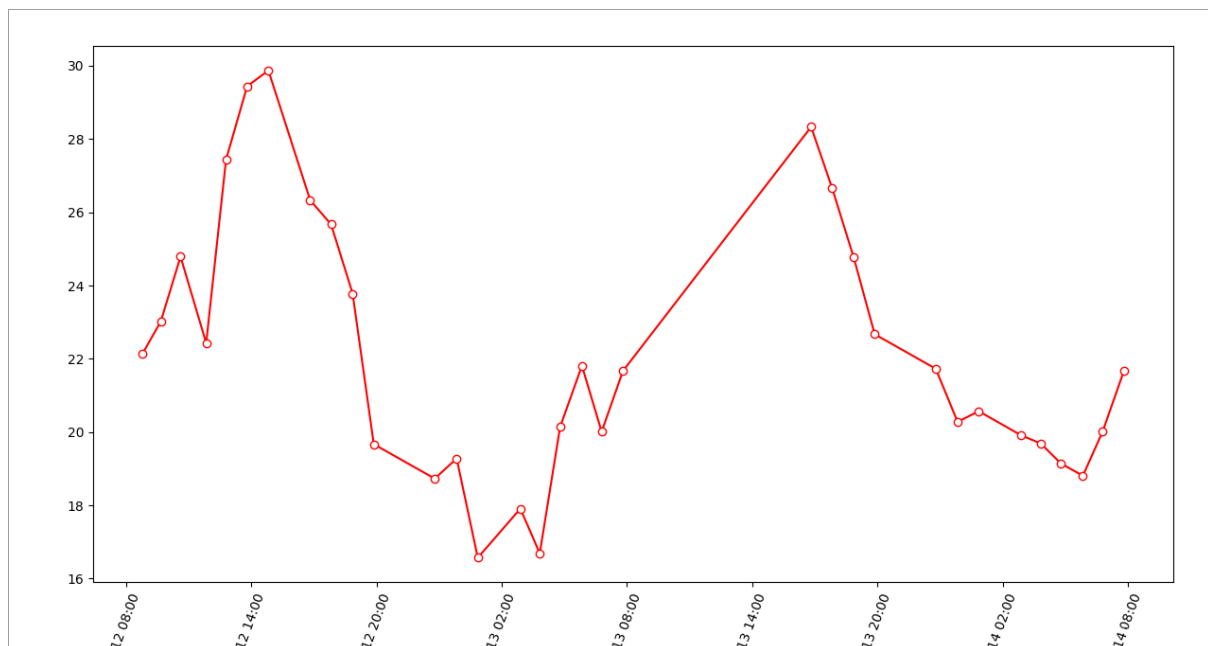


Figure: An example of a line graph.

The third scenario is the most complicated. The traditional line graph in the Cartesian coordinate system is unsuitable for displaying wind data because wind data contains wind angle. The wind data is analyzed using a polar diagram based on polar

coordinates. Our idea is to implement the polar diagram using the histogram function in NumPy. The first step is to obtain the `wind_speed` (wind speed) and `wind_deg` (wind direction) columns of the Variable. Next, divide 360 degrees into eight sectors, every 45 degrees, and assign the data to the corresponding sector. Histogram function will generate two data respectively, an array of sector data volume (wind speed within a sector) and an array of sector ranges (boundaries of each sector). After that, a `pi` array is created (specifying the outer circumference for the polar plot), the color is set, and the sector data volume array and the sector range array are called. Here, the outer circumference can be expanded to avoid overlapping the arc of the sector with the outer circle so that the user can better view it. Finally, call the `show` to draw. (See the figure below)

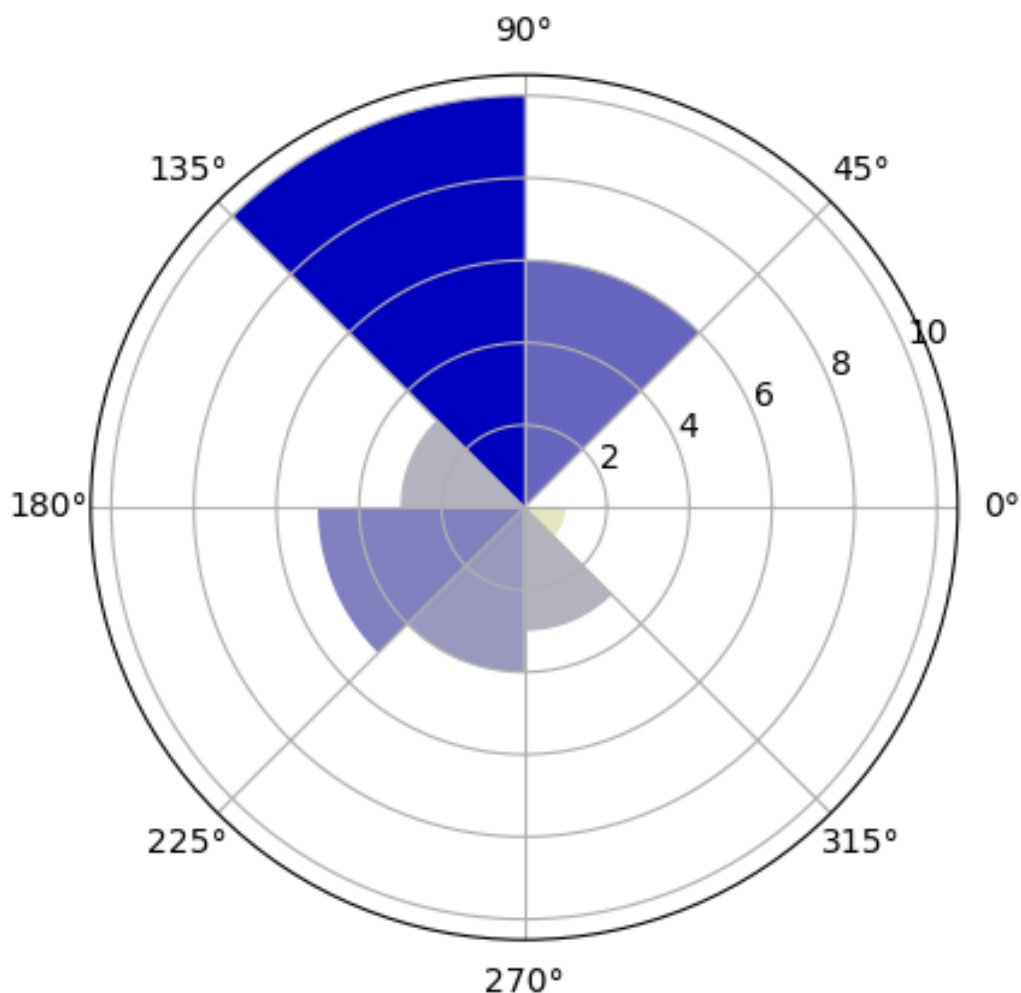


Figure: An example of a polar diagram.

## 7. Notable tradeoffs

Functionality vs. Size of the data

In data design, compared to the other file format, it is easier to process XML files. In the meantime, it is also more readable. However, the CSV file takes less memory compared to XML when it comes to storing the same amount of data. In this project we have relatively low requirements in the performance of data processing, therefore we choose size over functionality so we use csv file instead of XML files.

## 8. Notable risks

The risk of the current system comes from security and handling large amounts of visits.

All of the calculation of the system will be executed in a processing PC with execution Environment of Python 3.9.7 (see more detail in deployment diagram). It should satisfy the requirements in the previous document. (according to software requirements the system has to handle at least 10 user downloading at the same time) However, if the system meets a large amount of download requests the pc might not be able to handle them.

Another risk is the security, as we have talked about previously CSV is widely used in data storage in our system. It will make our system have the risk of being attacked by code injection. The projection contains a large amount of data, the attacker could hide malicious commands or functions in them via injection. These malicious commands might be executed and damage the interest of the user after they open the CSV files with Excel. These commands usually start with special symbols (+, -, @, = ...). 3 solutions are given regarding this situation: first, make sure that the units do not start with special symbols. Second, filter the content with special symbols. Third, transfer some of the content in the primitive input (for example adding a ' or / before "). These solutions will highly reduce the risk of injection attacks.

## 9. Milestones

ID	Date	Milestone	Description
M1	09/21	The Initial project plan by contracting the customers	The project should be selected and discussed between group members. A meeting is held between group and project customers and an agreement is reached with the customer that the project is to proceed. A supervisor will have been found for the project. All parties will have signed the project contract, based on the form provided on the CISC 498 web site. A project plan will have been created based on the project plan guidelines provided on the CISC 498 website.
M2	09/26-10/02	Define problem and processes by analyzing project	Analyze the scope of the project and identify the target users. Define the problem with these users as the core. At the same time, the issue needs to be broad enough to cover the project itself. In addition, by understanding the content of the project, we need to address the sequencing of the process, the range of the process, the resources required, and the results.
M3	10/03-10/09	Meet customers to identify aspects, limitations and acceptance criteria	Through the communication with the customer, understand the core and secondary parts of the project. Make a note of these two parts and distinguish them. Ask the customers about possible resource requirements for executing the project to mark the constraints of the relevant resources. In addition, recognition of acceptance criteria is achieved by understanding the customers' goals.
M4	10/10-10/19	Create requirements document/ Present Requirements	<p>Hold a meeting with customers and listen to their software requirements. We use a notebook to write down some key problems they are facing. We then come up with a feasible solution and discuss with customers. After several solutions are drawn, they will be documented into a google share doc in order to refer in the future. We also provide our customers with a low-fidelity prototype drawn from figma.io in order to give them an initial idea about the product and help us elicit further requirements.</p> <p>During the requirement elicitation, we will follow the bullet points listed in the requirement document, identifying: user group, context of operation, current processes, roles, functional requirements, non-functional requirements and academic glossary.</p>
M5	10/20-10/23	User interface design based on the	According to the user's situation, targeted to design the user interface. This requires understanding their goals, abilities, and preferences. In addition, attention should be paid to the

		user	simplicity and consistency of the interface during the design process. When designing the user interface, we should also ask customers for their opinions so that problems can be corrected in time. We will bring up a low-fidelity model used by customers to have an initial impression about what the product will look like.
M6	10/24-10/30	Communicate with customers to design prototype	At this stage, the prototype of the design should be a low-fidelity prototype. The initial prototype was designed using a combination of pen and paper and the website Figma.io. After that, we should present the prototype to customers. Customers will give us advice on how to improve the initial prototype design.
M7	10/31-11/06	Data design using diagram	First of all, ask the customers to know the meaning and purpose of each data. Draw diagrams such as ER diagrams to relate the relationships between the various data. After that, we need to discuss and decide on the format and specification of the data. At the same time, we should also decide how to query and import and export data based on relationships.
M8	11/07-11/13	Notable tradeoffs and risks discussing with customers	Review and summarize the trade-offs made during the design process, based on previous conversations with the customers. And analyze the reasons for these trade-offs. Write this down in text. At the same time, there are also various risks during designing. Again, we use the text to explain how these risks were overcome.
M9	11/14-11/23	Design document -Presentation	Based on the above external and internal designs, we summarize them by writing at least 20 pages of documentation. In addition to the user interface, prototype, data design, and trade-offs and risks, the documentation should also describe the project's usage scenario, feedback from customer, programming environment, algorithms, and so on. After that, we will divide the essay into five pieces as much as possible to prepare the content of the presentation.

## Reference

1. <https://www.queensu.ca/geographyandplanning/icelab/>
2. [http://archive.computerhistory.org/resources/text/algol/ACM\\_AlgoI\\_bulletin/1064048/frontmatter.pdf](http://archive.computerhistory.org/resources/text/algol/ACM_AlgoI_bulletin/1064048/frontmatter.pdf)