Design Document

Software Design

COMP.SE.110

Version history

Version	Date	Author	Description
0.1	10.02.2022	Entire team	First draft
0.2	15.03.2022	Entire Team	Mid term Document

Contents

1. Introduction

1.1 Purpose of the Document and Project.

The purpose of the document is to explain how the design has happened during the project planning process and in further implementation. The goal of the project is to build a stand-alone application for monitoring real-time data on greenhouse gases and comparing current data with historical data on greenhouse gases.

1.2 Product and Environment

The product is standalone application software which can enable monitoring. The team has decided to develop the product using the Java programming language. The project also uses JavaFX library for the most part of implementation. The team has decided to use NetBeans as the IDE and Gitlab for version control management.

1.3 Team Members

The project will be done by four team members Cecylia Borek, Gaurab Mahat, Roger Wanamo and Sai Polineni.

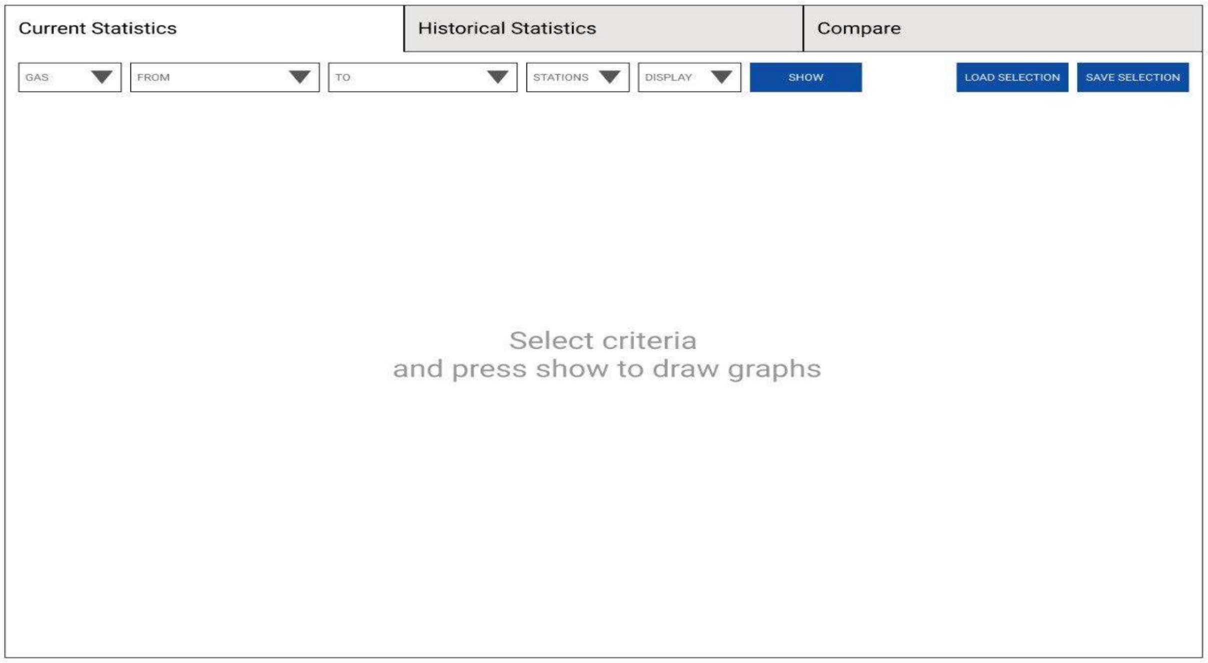
1.4 Related organization

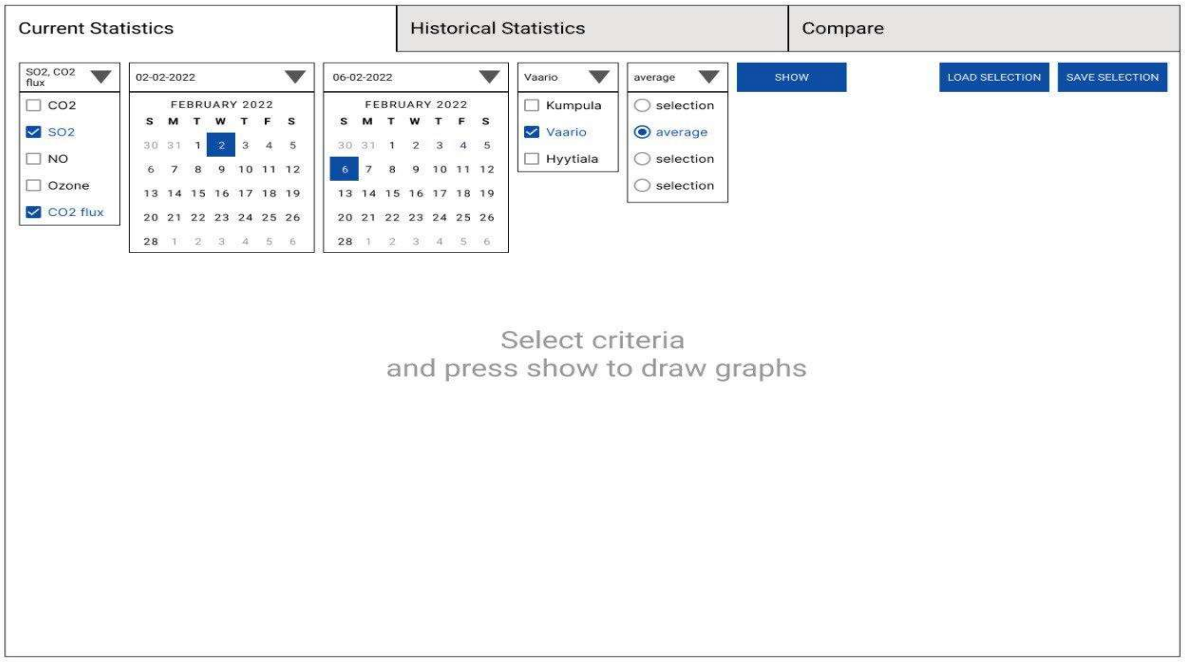
The project is assisted by Ville Heikkilä. Ville is one of the Teaching Assistants of the course. Responsible for guiding the project management towards the correct direction and providing assistance and answers when needed.

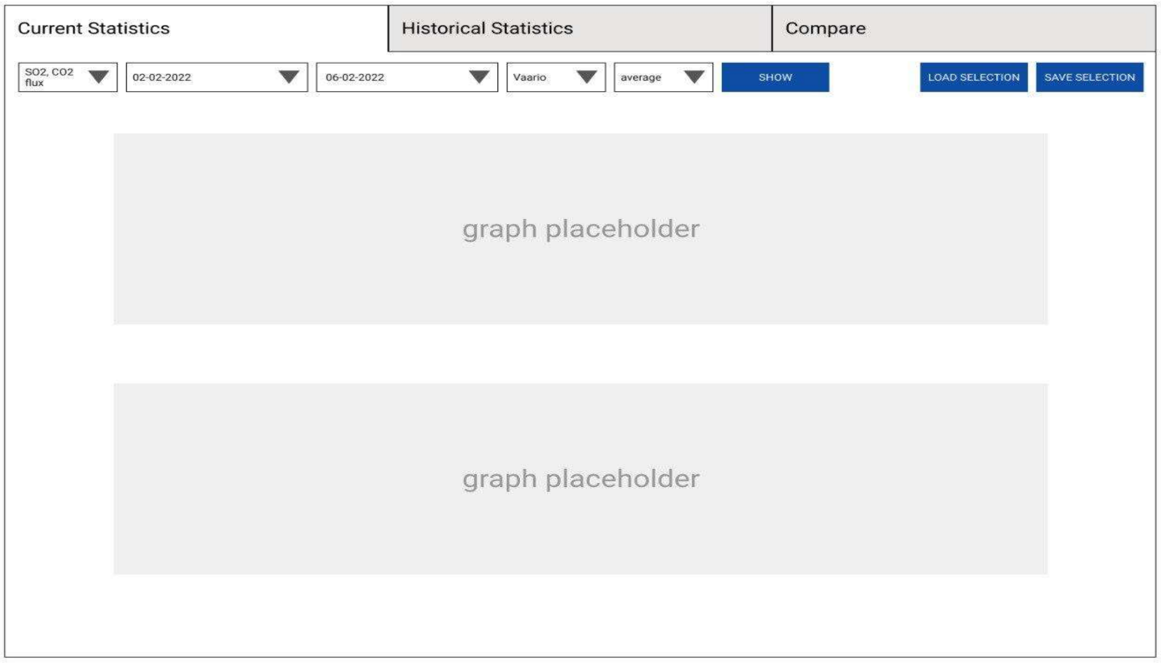
2. High Level Description

The Application will allow a user to monitor the current statistical data about the several types of greenhouse gases whose data has been taken from the SMEAR database and API. It also allows the user to watch the historical data and to compare the CO2 emissions. These operations are performed from the three tabs in the UI (User Interfaces) with the names Current statistics, Historical data, and the compare CO2.

Design Images:





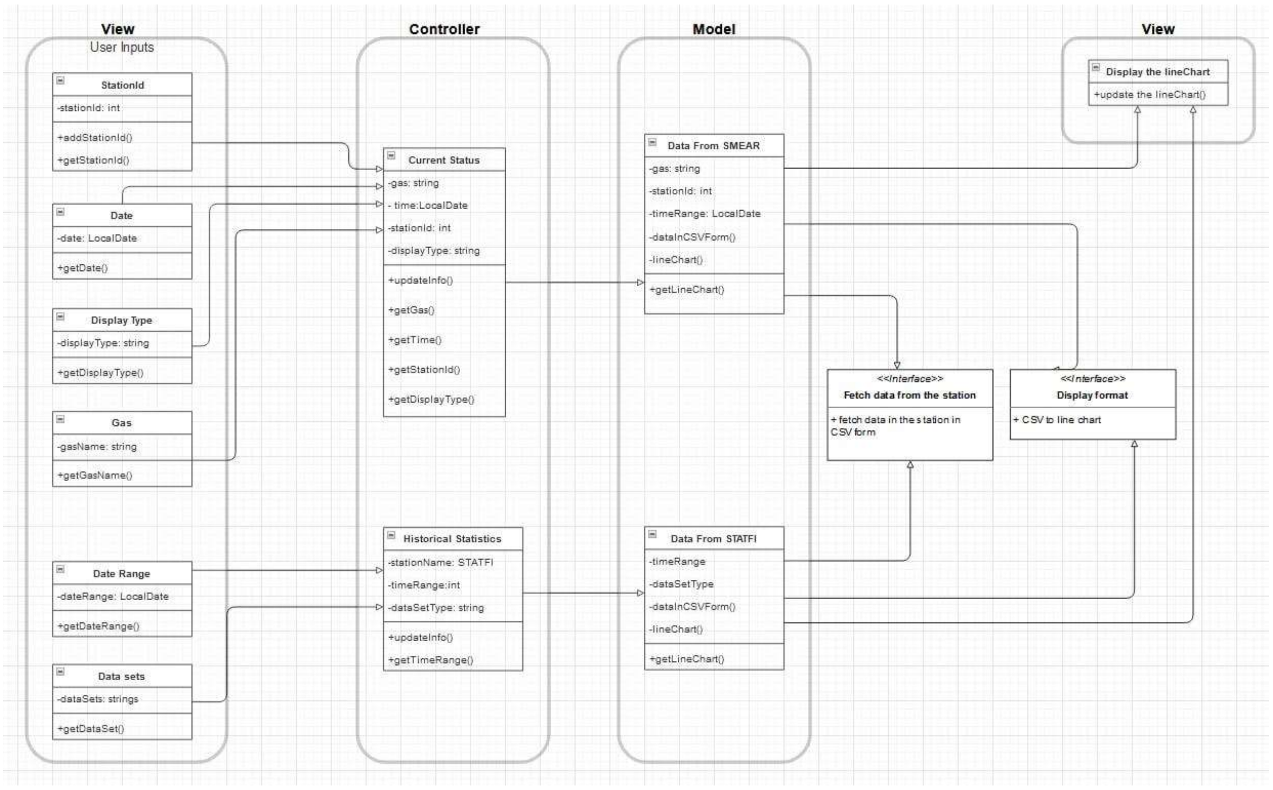


The current statistics tab will allow the user to monitor the current statistics about the different gas emissions. The Gas dropdown menu will enable the user to select a single or multiple gases at a time. The two date fields are used to select the dates with a time range in between them. The location menu contains all the current locations that can be monitored. Users can select multiple locations at a time. Users can view the graph by clicking the show button. The save selection button can enable the user to save their selection so that they can view their saved selections next time easily.

Similarly, the tabs for Historical statistics and the Compare are used to monitor the historical data from the STATFI and for the monitoring of the CO2 values.

Class Diagram:

The below class diagram will display the important classes and interfaces that are included in the project,



From the above class diagram, the View component consists of six classes. The stationid class will have the information about the station and methods to add a new station and get a stationid. The date class will contain the date attribute and a getDate method. The display type class holds the display type attribute and getDisplayType method. The Gas class will have the getName attribute and getGasName method. The Date range class will have the dateRange attribute and getDataRange method. Finally, the Data sets class will have datasets as an attribute and getDataSet method. These classes will serve as the user inputs in the application.

The Controller component will have mainly two classes, Current statistics and Historical statistics. The Current statistics class will draw the input from the Stationid, date, displaytype and gas classes and it also has the methods that are available in them. In addition, the updateInfo method is present to deal with the updates that are based on the users. The Historical statistics class will draw the inputs from the Date range and Data sets classes. It has an attribute stationName with STATFI, so that it uses the STATFI data for the monitoring of the historical data in this project.

The Model component consists of two classes, Data From SMEAR and Data from STATFI. The Data From SMEAR class will draw the input from the current statistics class, and it also has additional attributes such as dataInCSVFormat, lineChart and the method getLineChart. The Data From STATFI class will draw the inputs from the Historical statistics class and it has an additional attributes such as dataInCSVFormat, lineChart and the method getLineChart. These two classes communicates with the interfaces in fetching the data from the station and in displaying the CSV information into the line chart format.

Further Display the lineChart class will get the information from the Data from SMEAR and Data from STATFI classes and updates the latest line chart.

3. Boundaries and Interfaces

3.1 Information Flow

The information flows in the project based on the user selections. First the selections that are chosen by the user will flow as the commands to the query builder component, the query builder will convert the commands into suitable HTTP requests. The HTTP requests communicate with the backend using API interface and fetch the results. The results that are fetched in the form of JSON objects will be converted into the graphs by the data renderer component.

3.2 Components and Responsibilities

The general components that are used in the project are,

1. UI component: The UI component serves as the starting point for the whole application. It allows the user to interact and monitor the application. The UI is developed using FXML /CSS.

2. Controller Component: The Controller component interprets the commands that are given by the user, and it re-directs the commands to the logic component.

3. Application logic component: The component is used to handle the user commands that have been sent as the input.

4. Query Builder Component: The Query builder component creates the HTTP requests for the commands that are being sent from the application logic component.

5. API Interface Component: The API interface component makes the HTTP requests and communicates with the Backend. It also fetches the result data from the Backend.

6. JSON Interpreter: The JSON interpreter component is used to convert the JSON objects. These objects can be referred to the result records in this project.

7. Data Interpreter: The data interpreter component extracts the relevant data from the incoming data from the JSON interpreter component.

8. Data Renderer: The data renderer component turns the data into the graphs and those graphs can be shown.

9. System Hardware Component: The Component is used to store the graphs that the user wants to save in his device.

3.3 Packages and Interfaces used in the Implementation

1. fi.tuni.csgr Package: The package acts as the starting point for the whole project and contains all the implementation code in it.

2. Components Package: Contains customized reusable UI components.

   ChartView is a component that contains a linechart with title.

3. converters Package: The converters package contains three packages in it.

   3.1 helpers Package: The helpers package contains StationGas class. The methods getStation and getGas return the station and gas name whereas the boolean function equals compare the values with the object.

   3.2 json Package:

   The json package contains the interface JsonToResultConverter. The JsonToResultConverter interface has ResultList class which takes the Json data in the string format through convert method.

   The SmearJsonToResultConverter class implements JsonToResultConverter interface. It will call the StationGasSlugConverter class present in the slug package inside converters. It will return the conevrted json data in the form of ResultList.

   The ResultList implements the SGResult class. The getSGResult method will return the StationGas value.

   3.3 slug Package:

   The slug package contains StationGasSlugConverter class. It will map the string values with the station name variables and the gas variables.

4. managers.graphs Package:

   The DiffResult class in the managers.graphs package will add or remove the new key values.

   The GraphDataManager class is used to manage all the data being dispalyed in the graphs. The update() method is used to update the stored data with newly fetched results. The class conatains listeners for the functionalities such as storing all the result gases, storing station names, storing both gases and station names. The class also contains methods to removing or adding new gases, Removing or adding the station names for a particular gas.

5. network Package:

   The network package contains an abstract class Network which contains the GraphDataManager class from the managers.graphs package. The method GetData will contain the details of the station name, gas, from and to time.

   The SmearNetwork class inherits the Network class . The GetData method will fetch the data from the smear and updates the GraphDataManager class.

6. SMEAR Package:

   The SMEAR package contains the interface IGetData. The interface contains getDataInStringJson method that will be used to fetch the Json data from the SMEAR API in string format.

   GetTableVariable class contains the method getStationsCode which will return a string that will contain the tablevariable name of the gas corresponding to its station.

   DataFromSmear class implements the interface IGetData. It uses the method getDataInStringJson to fetch the data from the SMEAR API using HTTP request.

7. STATFI Package:

   The STATFI package contains the interface IGetData. The interface contains getDataInStringJson method that will be used to fetch the Json data from the STATFI API in string format.

   DataFromSTATFI class implements the interface IGetData. It uses the method getDataInStringJson to fetch the data from the STATFI API using HTTP POST request.

8. stationnames Package:

   The stationnames package contains the class stationNameMapping which map all the variable names of gases and the station names to the corresponding station.

9. utils Package:

   The utils package conatins datePickerUtils class that restricts selectabl e dates on datePicker to allow the dates from minDate selected to the maxDate.

   The MenuUtils class creates CustomMenuItems for ecah checkbox that is present in the itemList and add them to the sourceMenu. It returns the arraylist of all the selected MenuItems.

4. Reasonings

The decisions about the design have been made collectively by all the team members. The team members initially held a starting meeting for the project to discuss the approach and implementation of the project.

4.1 Self Evaluation:

1. Cecylia Borek: Created a package that will convert the Json data fetched from the SMEAR API into ResultList. Also created the GraphDataManger which manages all the data displayed in the graphs.

2. Gaurab Mahat: Created a package that will fetch data from the SMEAR. To fetch the data from the SMEAR the URL requires a table variable name that is based on the name of the gas and the station. A separate package is created that will be used to map the table variable name depending on the selected gas and station.

3. Roger Wanamo: Created the GUI and all things directly related to drawing things on the screen, including FXML the FXML controller, utils and components packages, as well as the CSS. The last part of my work, connecting the fetched data to the graphs on the screen, would have required some more time. There was a lot of unexpected problems, and the result is quite rushed. Some refactoring is needed in the next phase.

4. Sai Polineni: Tried to work with fetching the data from the STATFI API. Learned something about fetching and dealing with the APIs. Did the documentation for the Project.

Changes in the project:

The team has agreed on the initial prototype model that is designed in the first phase. There are no big changes from the implementation point of view of the project. There are some changes regarding the naming conventions of the classes and interfaces in the implementation from the design view. The team has moved from using the NetBeans IDE to the IntelliJ IDE for better version control.