Group assignment specification, Software Design, spring 2022

Overview

In this project, the student groups will **design** and **implement** a piece of software for monitoring real-time data on greenhouse gases, and comparing current data with historical data on greenhouse gases. Global warming and climate change are becoming more and more apparent in our everyday lives. One of the biggest factors behind climate change are greenhouse gases – carbon dioxide (CO_2) being the major one of them. Other gases, such as sulfur dioxide (SO_2) and nitrogen oxide (NO) also play their part.

There are statistics from several decades back on the average levels of greenhouse gases, namely CO_2 emissions and CO_2 intensity, per year. Now, the SMEAR project gives real time data from several measuring stations across Finland on these gases and other factors in our atmosphere.

For this group assignment, groups will 1) fetch and visualize real-time data on greenhouse gases using the SMEAR https://smear.avaa.csc.fi/ database and API https://smear-backend.rahtiapp.fi/swagger-ui/, 2) fetch and visualize statistics on historical data of greenhouse gases from the Statistics Finland (STATFI) database and API https://pxnet2.stat.fi/PXWeb/pxweb/en/ymp/ymp__taulukot/Kokodata.px/ , and 3) combine the real-time data with historical data to allow for easy comparison on current situation of greenhouse gases to past years. **There are four major use cases.**

1) Monitoring real-time data from SMEAR

The user is viewing current data provided by SMEAR. The user is able to select variables and choose what is shown in the visualization (gas, time period, measuring station, etc.).

2) Checking historical values from STATFI

The user is able to see a visualization of historical data on greenhouse gases based on the selected variable (dataset and time period).

3) Comparing the current situation to history.

The user is primarily viewing current data on CO_2 levels retrieved from SMEAR. The user then wants to see how the current levels compare to history. The user is able to fetch the averages from a desired interval from STATFI, and view these alongside the current data to see how the CO_2 levels now compare to those in previous years.

4) Breaking down averages

The user is primarily viewing statistical averages from the STATFI database. Looking at some year(s), they want to better understand how the collected CO_2 was gained over the selected year(s). They are able to choose SMEAR data from the given year to see how the trend of CO_2 evolves.

Requirements:

For use case 1):

- User can select which monitoring station’s (one or more) data they are interested in.
- User can select which greenhouse gas / data variable they are interested in (minimum options: CO_2, SO_2, NO_x)
- User can set a time period from which data is shown
- In addition to viewing raw data, user can ask for a minimum, maximum and average values of the selected data
    - For a given time period for a given station
    - For several stations for a given time
    - For several stations for a defined time period
- Not all measuring stations provide exact same data. However, the user interface should be user friendly in such a way that the user does not need to do “test-and-try" to see where the data exists. In other words: the application should only allow selecting data that is actually available.
- Error handling: you must efficiently prepare for errors in the data and handle null values.

For use case 2):

- The user can select a time range for viewing data from STATFI
- User selects one or more available datasets: CO_2 (in tonnes), CO_2 intensity, CO_2 indexed, or CO_2 intensity indexed (one or more).
- User is shown a visualization of the data

For use case 3):

- User is given a visualization of the CO_2, SO_2 and NO_x values from SMEAR for a time period specified by the user, for the station(s) selected by the user.
- The user can select data available from STATFI on historical averages
- The user can specify the time range (e.g. year 2010, or years 2000-2009) they are interested in.
- The historical values are shown alongside real data in a comparable way.

For use case 4):

- The user selects a time range for viewing data from STATFI
- User selects data from CO_2 (in tonnes), CO_2 intensity, CO_2 indexed, or CO_2 intensity indexed.
- User is shown a visualization of the data
- The user is able to choose real time data from SMEAR. **NOTE:** SMEAR data does not handle the same timespan as STATFI. You will need to check how long back data is retrievable from SMEAR and guide the user accordingly.
- Breakdown of the historical average to the selected year based on SMEAR data is visualized.

General:

- The user can save preferences (e.g. certain stations, certain time period in history, which greenhouse gas(es) from SMEAR, which statistics from STAT FI) and apply them when using the software later
- Design must be such that additional data sources could easily be added

Accepted programming languages are Java, C++, C# and Python.
Only stand-alone/desktop applications are allowed.

Web applications are **not** allowed.

Grading:

1. Prototype: 0-1p

Everyone who makes a submission *on time* and *according to instructions* will get 1 point.

2. Mid-term submission: 0-2p
   Evaluation criteria: self-evaluation, timeliness of submission, some actual code exists (work done on actual product based on prototype)
3. Design (document): 0-3p
   1. High level description of the design 0-1
   2. Describing components and their responsibilities: 0-1
   3. Interfaces (internal): 0-0.5
   4. Components' internal structure and functions: 0-0.5
4. Final submission (software): 0-7p (personal adjustment: –2 -- +1)
   1. Functionality -1 - 3
      - Possible bonus
   2. Division to clear areas of responsibility (helped in allocating work): 1
   3. Interfaces: 1
   4. Use of design patterns and other principles taught on course: 1.5
   5. Quality of documentation, Readibility of code, Other possible factors: 0.5p
   6. Timeliness: submitted on time. If a group has a need for extension (for a good reason), contact your TA and discuss. Late submissions will lead to decreased points. TAs will decide how many points will be taken based on how much the deadline was extended, what kind of grounds there were for being late, and overall status of the submission.

5. Peer review: 0-2p (for reviewers)

Bonus points may be acquired from the following:

• Enabling saving the visualizations (graphs/plots/tables/other) as images.

• Allowing the user to choose between several plotting options.

• Adding a third data source in a meaningful way (e.g., Fingrid or FMI)

Bonus requirements can be implemented to compensate for some lacking functionalities in the basic requirements. Definitive points for bonus requirements are not given, as bonus requirements cannot be used to compensate for poor design.

If the sum of points is larger than 15 points (meaning that the group has gathered full points for everything and still done bonus requirements in addition), the maximum points awarded will still be limited to 15p, meaning that the group assignment has an equal effect on the course grade as the exam.

Deadlines and timeline:

Submission deadline for Prototype: 20.02. Prototype evaluation. Two items to be submitted:

1) **Software prototype.** The purpose is to get the group to start thinking about implementation and to illustrate how the group has understood the requirements (helps discussions with the TA). There are several acceptable options for delivery:

Option 1: A well-thought sketch of the UI, showcasing all the requirements, including transitions between selections (using, e.g., Figma or some other method of achieving a high-quality sketch)

Option 2: A rough implementation of the UI, with something that ”functions and moves”. Actual code but made with minimal effort. *Be prepared to throw this away and start implementation of actual application from scratch!*

Option 3: Something between Options 1 & 2: quickly made implementation of UI elements, not necessarily any functionality, but clearly showing how requirements would be implemented.

2) **A design document**, describing the structure and most important components and interfaces of the software. No official template will be given, it is advised to include a figure or chart (or couple) of some kind.

22.02. – 25.02. Group meetings with TAs, a demo of the prototype and discussion on group’s understanding and plans for implementing requirements, discussion on documentation.

Submission deadline for Mid-term: 20.03. Mid-term evaluation. Two items to be submitted:

1. Software. At this point some of the functionalities should be properly implemented.
2. An updated design documentation. The applicability of the original design should have been assessed based on implemented functions. The design documentation must thus include a self-evaluation of the proposed solution and suggestions of required changes (how to refactor the software).

22.03. – 25.03. Group meetings with TAs. Discussing and giving a demo of the current version of the software. Making plans for implementing the remaining requirements. This is a point to particularly discuss what should be changed in the structure of the software.

Submission deadline for Final submission: **22.04.** Final submission. Two items to be submitted:

1. Software. Everything has been implemented.

2. An updated, final version of the design document. One particular part to document is how the software has been refactored during the implementation process, and how refactoring has succeeded.

Final submission includes submitting these both to your TA, and to the group who is reviewing your work as part of peer review! Final submission is not considered done if you have not given your work to your peer group.

25.04. – 29.04. Meetings with the TA. Demo and discussion, particularly what went smoothly and where did the group find challenges.

Submission deadline: 25.04. – 29.04. TA meeting time. Peer review.

Each group is assigned another group whose code and documentation they will review and comment. Reviewers will get points.

General guidelines for submissions:

All submissions (to a specific deliverable, i.e., proto, midterm or final) will be made to the groups' Git repositories, and commits relevant for the deliverable should be marked with tags such as "Proto deliverable, data retrieval", or with some other clear tag message. This will help the TAs' job to check that all submissions have been made.

Groups are required to write a clear README file, noting the environment required to run the software, and what files need to be executed and for which purpose. The instructions must be so that the TA is able to download all necessary libraries and compile and execute the code on his/her own computer. For example "As an environment we use a ready installed Qt Creator with VPN… For fetching data you must run the main_data_gathering.cpp file under folder Data_fetch."

For the group meetings the TAs will create Doodle polls, from which each group will reserve themselves a meeting time.

In the meetings the group will present the documentation required for the submission, give a demo of their implementation, and discuss their solutions.

Remember to have all documentation and code/commits ready to show in the meeting.

More specific instructions for each submission will be given closer to the deadlines.