Group project description, Software Design, spring 2021

In this project, the student groups will **design** and **implement** a piece of software for monitoring electricity consumption in parallel with weather forecast. How much electricity is used has a lot to do with the weather – in addition to simply cranking up the heat when the weather is cold, also windy weather will often lead to adjusting the heating. Additionally, there are nowadays several power sources (solar power, wind power) that directly depend on the weather.

In this project, students will utilize power data from Fingrid https://data.fingrid.fi/open-data-api/, https://data.fingrid.fi/en/dataset (particularly: https://data.fingrid.fi/en/dataset?groups=load-and-generation and https://data.fingrid.fi/en/dataset?groups=state-of-power-system), and weather data from FMI https://en.ilmatieteenlaitos.fi/open-data-manual. Groups will build an application that allows consumers to see both power and weather data individually and combined in an intuitive way to show how weather has affected both supply of certain power sources and the demand of electricity.

Students are allowed and expected to use their own creativity in specifying the functionalities and look of the application. The purpose is that each group will discuss internally, and the set of functionalities that will be implemented will be agreed upon in the first meeting with the group's assigned teaching assistant (TA). The requirements below leave room for interpretation on purpose. It is up to the groups to implement based on their interpretation and discuss with their assigned TA.

The basic requirements below are made of practical requirements and functional requirements. There are also some bonus requirements, which are not compulsory to pass the course, but implementing them may earn extra points.

Practical requirements:

- Sensible use of design patterns and other principles taught on the course
- Documentation
- Readibility of code
- Sensible use of version control
- Interface design and implementation

Functional requirements (basic requirements):

- The user is able to choose what kind of electricity data he/she wants to see (from a set of options) and will be shown a visualization of the selection. Visualizations should include graphs or plots (or some self-developed visualizations) at least for time series type data. Tables can be used when they are more suitable.
    - Within the visualization window, the user can adjust the parameters of the visualization (timeline, power source, etc.)
    - User must be able to see at least the following and combine them in some way:
        - Electricity consumption in Finland
        - Electricity consumption forecast for the next 24 hours.
        - A tentative production prediction for the next 24 hours as hourly energy
        - Electricity production in Finland
        - Wind power production forecast
        - Nuclear power production
        - Hydro power production

    - User must be able to request calculations and visualization of percentages of different power forms (hydro, wind, nuclear)

- The user is able to choose weather data (from a set of options) and will be shown a visualization of the selection. Visualizations should include graphs or plots (or some self-developed visualizations) at least for time series type data. Tables can be used when they are most suitable.
    - Within the visualization window, the user can adjust the parameters of the visualization (timeline, location, selected weather information)
    - The user must be able to see at least the following and combine them in some way:
        - Temperature
        - Observed wind
        - Observed cloudiness
        - Predicted wind
        - Predicted temperature
    - The user must be able to request calculations and visualization on average temperature at certain location in certain month
    - The user must be able to request calculations and visualization on average maximum and average minimum temperature at certain location in certain month

- The user is able to combine electricity and weather data into one window, particularly, the user must be able to:
    - Combine current weather forecast, current energy production forecast, and current energy consumption forecast into one view.

- **<updated>** The user is able to initiate **large-scale history** data collection.
    - The user can initialize data collection for **the** real time data provided by Fingrid, and set a timeline for collecting data (e.g., **previous 180 days**). The user will be able to visualize the collected data. **The purpose is that the application must be able to handle also very large quantities of data (such as the Fingrid data with data points for every 3 minutes, fetched from a long period of time).**

- The user is able to save certain data sets and produce visualizations. For example: the user fetches a visualization of energy production forecast for the next 24 hours, and saves the data. Two days later the user wishes to see how accurate the forecast was, so he/she is able to fetch the saved forecast and compare it with the real production of the same time period.

- The user is able to save preferences for producing visualizations (e.g., a specific source of power combined with specific weather data), and fetching those preferences will produce a visualization using the most recent data with the given parameters

- The design must be such that further data sources, e.g., Nordpool, or additional data from existing sources could be easily added

Bonus requirements:

- Saving the visualizations (graphs/plots/tables/other) as images.
- The visualization window is updated in (near) real time, if user selection contains data that is updated in shorter than hourly intervals
- User can choose between several plotting options.
- Implementing a mobile alternative.

The group can utilize any object-oriented language, though preferable ones are C++, Java, C# or Python. If you decide on something else than C++, make sure to agree on the chosen language and frameworks, and how to submit and inspect your software together with your assigned TA before starting implementation.

Grading:

- Prototype: 0-1p

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

- Mid-term submission: 0-2p

  Most important aspect is self-evaluation.

- Design (document): 0-3p
  - High level description of the design 0-1
  - Describing components and their responsibilities: 0-1
  - Interfaces (internal): 0-0.5
  - Components' internal structure and functions: 0-0.5
- Final submission (software): 0-7p (personal subtraction 0-2)
  - Functionality -1..3
    - Possible bonus
  - Division to clear areas of responsibility (helped in allocating work): 1
  - Interfaces: 1
  - Use of design patterns and other principles taught on course: 1.5
  - Quality of documentation, Readibility of code, Other possible factors: 0.5p
- Peer review: 0-2p (for reviewers)

If the sum of points is larger than 15 points (meaning that the group has gathered full points for everything and additionally completed bonus requirements), 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.

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.

Deadlines and timeline:

Submission deadline for Prototype: 21.02. Prototype evaluation. Two items to be submitted:

1) Software prototype, with some kind of implementation of the UI, with something that ”functions and moves”. Something made with minimal effort, to get the group to start implementation and to illustrate how the group has understood the requirements (helps discussions with the TA).

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.

23.02 – 26.02. 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: 21.03. 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).

23.03. – 26.03. 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: **18.04.** 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.

20.04 – 23.04. Meetings with the TA. Demo and discussion, particularly what went smoothly and where did the group find challenges.

Submission deadline: 26.04. 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 midterm evaluations and final submission will be given closer to the deadlines.