Requirements specification (Team 2)

Purpose and Overview

- ### Intended Audience and Intended Use:

This SRS[1] will be available for the Team 2 of the SSP[2] of the 2022 SE[3] course as a guideline for a common idea of the project. It will also be available for other team leaders and the project leader for seeing what our work is about and propose suggestions for further improvements of the project.

- ### Product Scope:

Our goal is to create a software which would calculate the most optimal location and pathway on the roof for the local fire brigade for security necessities in case of fire. This will be done according to the Lithuania's fire security regulations and the data provided by the Team 1. ***Note:*** Solar panels will not be allowed to be placed on the calculated zone.

- ### Possible Risks / Complications:

There are a couple of possible risks that might occur during the task:

1. Being late on the deadline because Team 1 will not have provided their data result in time which is needed to create relevant calculations for our part.
2. Our calculated result might not satisfy the needs of Team 3 – we will need to communicate in order to create a software that will make their work easier.

- ### Definitions and Acronyms:

[1]SRS – Software Requirements Specification.
[2]SSP – Solar System Project
[3]SE – Software Engineering
[4]TL – Team Leader
[5]PL – Project Leader

Specific Requirements

- ### Functional requirements:

Functional requirements are the type of requirements that the user specifically demands as basic facilities that the system should offer.

For our team the goal is to create a software which would calculate the best areas for dire ventilation setbacks or pathways on a building. It must comply with the requirements of the fire code and should maximize the area that could be used to put solar panels on.

- System automatically calculates the most optimal location and pathway on the roof for the local fire brigade for security necessities in case of fire.
- After the error the system should ask to check the inputted data
- Given data should be easily accessed and used for other teams.

- ### External Interface / other: *MAYBE NECESSARY*

- ### Quality attributes (non-functional req.):

Availability – software should execute tasks it is assigned to perform. Users should be able to use it without any delays or problems.

Correctness – software should be able to calculate the results correctly when given the expected type of data. The correct results should correspond to all necessary safety regulations, be the most optimal of all possible solutions and consistent.

Compatibility – this software should work consistently with the software from teams 1 and 3. Softwares should be able to exchange and reuse data easily and without any issues.

Usability – software should be easily used and work correctly if used for the expected purpose. Documentation describing its correct usage should be provided and be clearly understandable.

Functionality – software's functionality should correspond to all functional requirements mentioned above.

Reliability – software should not break in case of incorrectly passed data or other user faults and be available whenever needed.

Security – it should be guaranteed that only authorized users can access the software.

Implementation

Plan of action:

1. We are going to gather all necessary information on what we are going to get from team 1 and what team 3 needs and expects from us.
2. Then our team will create data for testing, similar to what we are going to get from team 1.
3. Create algorithm for all necessary calculations.
4. Refine algorithm to suite quality attributes and functional requirements.
5. Test algorithm with testing data that we made before.
6. Visualize algorithm, for better understanding.
7. Latest and most polished data from team 1 will be implemented with our algorithm.
8. Repeat step 4.
9. Export created results and combine them with team 3.
10. Get feedback from team 3 to improve our part and overall project.

Job distribution:

As our team is planning to work with *Top-Down* approach, our final picture will be made from several components, those components break apart in more specific components and so on. That way we can split smallest components with team members, and everyone can be involved in all parts of the final picture. All four of our team members planning to distribute 25% each.