VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Software Engineering | Project Requirements specification

Solar design tool

Area 2

Done by:

Martin Martijan
Gerda Zykutė
Norvydas Martinka
Gustas Vasilevič

Supervisors:

Lekt. Virgilijus Krinickij
Lekt. Gediminas Rimša

Vilnius
2022

Contents

Definitions and Acronyms

SRS_1 - Software Requirements Specification.
SSP_2 – Solar System Project
SE_3 – Software Engineering
TL_4 – Team Leader
PL_5 – Project Leader

Purpose

Intended Audience and Intended Use:
This SRS_1 serves as a guideline for the Team 2 of the SSP_2 of the 2022 SE_3 course as a guideline for a common idea of the part of the project. It describes the project's scope and functionality. This document describes the primary state of scope and functionalities and it will be modified in the duration of development. It can be used by other teams and the project leader for coordination of the project.

High level overview

The goal of our team's work is to create an API which would take JSON input, representing roof measurements and coordinates and calculate the best areas for dire ventilation setbacks and pathways on the building, returning the calculated data in JSON format. The output must comply with the requirements of the fire code and should maximise the area of solar panel placement.

Functional requirements

- The API should accept JSON input in an expected format, which represents coordinates and measurements of the roof.
- In case of an incorrect input, failed authorisation or other errors, the API should provide necessary information for the user to understand the error and be able to fix it if possible.
- The API should automatically calculate the most optimal location and pathway on the roof for security purposes in case of a fire.
- The API should format the calculated data back into the JSON object.
- The API should be able to send the JSON output to another API.
- The calculated measurements should correspond to all Lithuania's fire security regulations and maximise the area of solar panel placement on the roof.
- The API should be able to detect obstacles (e.g. windows) and be able to avoid them in calculations.
- The API should be able to make calculations for any shape and size of the roof.

Quality attributes

Availability - the API should be available at all times and should not have any downtime. It should not break in case of incorrectly passed data or other user faults.

Reliability - the API should be secure to use. It should have input filtering and checking methods and should ensure that only the expected input is used and that the input comes from secure and known source.

Security - the API should be accessible to use only from known APIs and should not accept unauthorised access from elsewhere. It should implement input filtering and checking methods and should ensure that the provided input is safe to use. Sensitive information should not be passed to output as plain text and should be encoded.

Usability - all the functionality mentioned in functional requirements section above should be working as expected when the API is used correctly. It should provide consistent and correct results, which meet all the requirements and regulations of the fire department. Documentation describing its correct usage should be provided and clearly understandable.

Implementation

Week 1 (09.05-09.11)
Participating in the introductory lecture to the project. Dividing into teams.

Week 2 (09.12-09.18)
Choosing parts of the project to work on. Familiarising with the given part.

1) What was done?
We had a team meeting on Wednesday discussing the content of the Requirement Specification. We made notes what should we write about and what is the purpose of it. After that each of us wrote a particular part of the Requirements Specification and team leader will have the draft published by Thursday night. We started discussing what kind of software will we use and how we will get the result, however we didn't really think of anything specific.

2)What issues/blockers did we face?
We postponed the meeting that we eventually had on Wednesday for 2 times - some students were unable to come to university on Monday or Tuesday and we didn't really want to have a meeting online. We also realised that we don't know who will be in TEAM3 so it's not clear to whom we need to talk in order to know what we will need to create.

3)Goal for next week:

- Finish the Requirements Specification;
- Create an understanding of which tools will we use;
- Distribute our roles in creating software;
- Ideally it would be great to understand what kind of information we will receive from TEAM1 and in what appearance it should be delivered to TEAM3;

4)Status: GREEN

Week 3 (09.19-09.25)
Preparing a draft for requirements specification. Creating and starting to use Git repository.

1) What was done?
We finished the content of our Requirement Specifications. Fixed issues we had last week and added more content. Started writing it in Latex.
We talked with other teams and created a mutual agreement of working with Java, so it is easier to merge code and parse results between teams.
We decided that we will create an API to implement our task.
We talked to TEAM1 and asked them to give us a JSON file as their result.
We had a particular success in distributing our jobs during the creation of Requirement Specification everyone successfully finished their responsibility and we managed to finish everything a week before deadline

2) What issues/blockers did we face?

Deciding what we want to build. Communicating with everyone and mutually agreeing that we are all working with Java. Converting our Specification into Latex :) Was the hardest problem so far

3) Goal for next week:

- Converting Specification document into Latex
- Deciding what exactly we need in a JSON file from TEAM1
- Generate ideas on how we will split TEAM3 work

4) Status: GREEN

Week 4 (09.26-10.02)
Completing the requirements specification. Agreements on the main technical points (programming language, type of software, etc).

1) What was done?
Polished our requirement specification
Found a mathematical solution to our task (future logic of the program)
Exchanged information with team 1 and received a demo JSON from them

2) What issues/blockers did we face?
Again, working with latex and converting everything according to standards
Going through mathematics of 3d space and vectors to find how we will solve our task

3) Goal for next week:

- Start coding (at least basic main method and maybe draft of other classes)
- Get more into the contents of JSON, how and which info we will use

4) Status: GREEN

Week 5 (10.03-10.09)
Initialising an empty project for the team, preparing the environment to work on, installing the required packages and libraries. Generating ideas for the calculation algorithm.

Week 6 (10.10-10.16)
Generating ideas for the calculation algorithm. Making the final decision regarding its logic.

Week 7 (10.17-10.23)
Creation of algorithm and testing it with test data locally.

Week 8 (10.24-10.30)
Finalise creation of algorithm. Preparing methods to format data in a way that is expected from team 3.

Week 9 (10.31-11.06)
Implementing input collection and output sending methods.

Week 10 (11.07-11.13)
Testing algorithm and data formatting together. Creating API security testing methods.

Week 11 (11.14-11.20)

Testing the API with real data from team 1. Connecting the API with the API from team 3.

Week 12 (11.21-11.27)
Final testing and completion of the production-ready API

Versions

Version 0.1.0 - the API will be able to find the ridge points coordinates and rake points to which they connect and present them

Version 0.2.0 - the API will be able to calculate fire ventilation setbacks for a roof with one ridge and no obstacles, using mock data.

Version 0.3.0 - the API will be able to calculate fire ventilation setbacks and pathways for a roof with one ridge and no obstacles, using mock data.

Version 0.4.0 - the API will be able to calculate fire ventilation setbacks for a roof with multiple ridges and no obstacles, using mock data.

Version 0.5.0 - the API will be able to calculate fire ventilation setbacks and pathways for a roof with multiple ridges and no obstacles, using mock data.

Version 0.6.0 - the API will be able to calculate fire ventilation setbacks and pathways for a roof with multiple ridges and obstacles, using mock data.

Version 0.7.0 - the API will be able to calculate fire ventilation setbacks and pathways for a roof with multiple ridges and no obstacles, using real data received from team 1 and send required data to team 3.