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 application that 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 calculations in data array format. The output must comply with the requirements of the fire code and should maximize the area of solar panel placement.

Functional requirements

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

Quality attributes

Availability - the application 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 application 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 a secure and known source.

Security - the application should be accessible to use only from known applications and should not accept unauthorized 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 the functional requirements section above should be working as expected when the application 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)
Introducing and familiarising ourselves with the task.

Week 3 (09.19-09.25)

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 4 (09.26-10-02)

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 5 (10.03-10.09)

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 6 (10.10-10.16)

1) What was done?
Analyzed the JSON that team1 provided Exchanged ideas on how to start coding - the first step will be to find the ridges from the file and system.out its points. Created an empty main class and some empty classes for implementing backend part. Setup new github branches and jiro

2) What issues/blockers did we face?
Analyzing the JSON and understanding from where we need to begin. Starting to know how to practically use github and jiro

3) Goal for next week:

- Create a code to find a ridge from the JSON
- Try to get used to using git and jiro, as well as try TDD method while coding

4) Status: GREEN

Week 7 (10.17-10.23)

1) What was done?
Reworked idea of JSON (simplified) Started coding and finding the ridge from a mock JSON Applied test driven development from the beginning (test if points are being read correctly) Template for Technical Specification

2) What issues/blockers did we face?
Talking to each other and explaining that we need less input to work more efficient and easier.

3) Goal for next week:

- Have a more or less finished Technical Specification
- Successfully find ridge points from simplified JSON

4) Status: GREEN

Week 8 (10.24-10.30)

1) What was done?
Haven't found ridge points, but advanced in JSON digesting Created some UMLs for our Technical Specification Almost added all the content needed to our Technical Specification

2) What issues/blockers did we face?
We had some issues while trying to read JSON, getting familiar to nuanses of object mapping Deployment diagram - some troubles

3) Goal for next week:

- Finish technical specification
- Finish reading the JSON and mapping it into classes (java objects)

4) Status: YELLOW for coding GREEN for deadline requirements (Technical Specification)

Week 9 (10.31-11.06)

1) What was done?
Finished Technical Specification Created a program to find setback area for a simple roof using mock data

2) What issues/blockers did we face?
Finishing the Technical Specification (diagrams especially) The program will not work if RAKE points are in bad order: first should be the top point (that forms the RIDGE). Will need to solve that later. Still haven't found the way to calculate the pathway to our setback area.

3) Goal for next week:

- Apply tests to the current program
- Read JSON data and then use the program to solve our problem

4) Status: GREEN (advanced in coding very much, the hardest job is probably done)

Week 10 (11.07-11.13)

1) What was done?
Uploaded spine code to github, everybody successfully pulled it and it works for them. Will continue building on this backend part: apply tests and json reading

2) What issues/blockers did we face?
Couldn't start building automated tests on the project from github (some kind of folder problems)

3) Goal for next week:

- Apply tests to the current program
- Read JSON data and then use the program to solve our problem

4) Status: YELLOW (haven't progressed in coding itself)

Week 11 (11.14-11.20)

1) What was done?
Added automated test to the program Fixed technical specification Prepared for mid-term presentation

2) What issues/blockers did we face?
Adding required test dependancies to the existing program on github

3) Goal for next week:

- More automated tests
- Start working on deeper code complexity (for more complicated roofs)
- Read JSON data and then use the program to solve our problem

4) Status: GREEN

Week 12 (11.21-11.27)

1) What was done?
Merged area 2 and area 4 Refactored main Json reader almost fully implemented

2) What issues/blockers did we face?
Build tools conflict while merging

3) Goal for next week:

- More automated tests
- Start working on deeper code complexity (for more complicated roofs)

4) Status: GREEN

Week 13 (11.28-12.04)

1) What was done?
Json reader fully implemented (no need in mock data) Backend logic error fix (setback lines wouldn't be calculated) Refactoring for better readability and usability No conflicts between AREA2 and AREA4, we use the same json reader.

2) What issues/blockers did we face?
When mock data is deleted, calculations wouldn't work. mainly needed to fix string equality in if statements: from == to .equals

3) Goal for next week:

- Try AREA1 json
- Think how to find a paralel line to ridge for complicated roofs

4) Status: YELLOW

Week 14 (12.05-12.11)

1) What was done?
Tried area1 JSON Found a way to find a paralel line to ridge for complicated roofs

2) What issues/blockers did we face?
Area1 json is not what we expected

3) Goal for next week:

- Update documentations
- Implement paralel line logic (maybe)

4) Status: YELLOW

Week 15 (12.12-12.18)

1) What was done?
Updated specifications Merged all areas Prototype works for a simple roof

2) What issues/blockers did we face?
The prototype of final product works buggy with a complicated roof - scenario won't be implemented in time.

3) Goal for next week:

- Find the best way to present and describe the work of our area

4) Status: YELLOW (we didn't finish the complicated roof scenarios but its working for a simple one)

Versions

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

Version 0.2.0 - the application 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 application 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 application 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 application 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 application 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 application 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.