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

Software Engineering | Project Requirement specification

Solar panel project

Area 1

Done by:

Arnas Vidžiūnas
Mindaugas Neseckas
Aleksandras Kuznecovas
Valentinas Straigis
Tomas Jefimovas

Supervisors:

Lekt. Virgilijus Krinickij
Lekt. Gediminas Rimša

Vilnius
2022

Contents

Purpose

The main goal of our team is to implement logic that takes a 3D model XML file produced by RoofOrders company, reads it, and extracts the information into JSON format needed in the further solar system design process steps.
The extracted data with our algorithm will be used to calculate areas for fire safety. Also our team's work will be essential for team 3's part of the project as they will calculate all the possible areas to install solar panels avoiding obstacle's locations that will be provided by our team.
Also, our data is going to be used to determine the location where the railings of the solar panels will be installed. All in all, our team's and this whole project will revolutionize the solar panel installation process as it is going to automate the planning phase of the solar panel installment.

Audience and use of this document

This Software Requirements Specification will be as a guideline for the Team 1 of the Solar Panel project of the 2022 Software Engineering course. This document will be available for the whole 3rd group and lecturers of the Software Engineering course. It will allow our supervisors to follow what state our project is in at any given time and give the chance to propose ideas on what could be improved and which aspects need more polishing.

High level overview

The goal of our team's work is to create an API, which would take the XML file and extract needed data to Area No.2 in JSON format. All of the other teams work is depended on our extracted data.

- **Version 0.1** - the package will be able to read any XML file and outputs everything to a JSON file.
- **Version 0.2** - the package will be able to check the XML file and if it is in the wrong format it prompts an error.
- **Version 0.3** - the package checks if there are not any malicious scripts. If there are any - the program exits.
- **Version 0.4** - the package checks if the provided data is correct, checks for certain assets.
- **Version 0.5** - the package reads the XML file, assigns needed data to keys and values and outputs it into a JSON file.
- **Version 0.6** - the package is able to work around the basic roofs.
- **Version 1.0** - the package is able to work around with any kind of roof. It checks if it is in the right XML file, and finds any malicious scripts. Assign data to keys and values and output it into a JSON file.

Our risks and complications:

1. Lack of communication with other areas. Possible that the other areas won't be responding to any messages and Area No.1 won't have any information about what data has to be provided for the project to go fluently.
2. Illnesses of team members. It is the possibility that if one or more team members get sick, the work process could be prolonged.
3. Provided data is not suitable for other areas. Possible scenario: Area No.2/3/4/5 gave us information on what data they need, but in reality they/we mixed something up, and wrong data was provided.
4. Problems with our chosen algorithm. In some unique cases wrong data is assigned to variables and importation of the 3D model has some inaccuracies.

Functional requirements

- The package will be able to receive and understand a 3D model of any roof variation.
- The package will check the provided data, if it is supplied in the XML extension. After that, the data will be cycled through the content to check if the data in the file is provided in XML format.
- In case of incorrect data, the module will prompt an error. The user will be asked to fix, rewrite or simply change the file.
- The package takes, and is able to interpret data in XML (Extensible Markup Language) format.
- The package will efficiently extract all of the necessary data from the provided XML file.
- The package will convert the data into the JSON (JavaScript Object Notation) file format. It will be the designated data format that will be given to the next team of the production phase.
- The measurements that carry the most value will be extracted from the data, as measurements of smoke shafts, angles of the individual roof sides and etc.
- The package will send the JSON output to other teams APIs.
- The package will provide fast and quality delivery of the extracted information for following calculations.

Quality attributes

Security - data in the package would not be able to be accessed by 3rd parties. The package will check if there are not any malicious scripts.

Usability - software will be easy to use, won't have any complex functionalities. Additional instructions on how to use the program will be provided.

Availability - software will be able to be executed at any given time without any delays or other problems.

Reliability - program will prompt errors if the wrong data file is imported and will automatically close if malicious scripts are detected.

Implementation

1. **Week 1 (09.05-09.11)**
   Participating in the first introductory lecture/workshop to the project. Dividing into teams and getting to know the project requirements.

2. **Week 2 (09.12-09.18)**
   Dividing into different areas, selecting teams and project leaders. Discussing what programming language will be used.

3. **Week 3 (09.19-09.25)**
   The goal of this week is to setup our working environment. We decided that our module is recommended to use with Python 3.9 version.

4. **Week 4 (09.26-10.02)**
   Set up the Jira project management software. Learn to use LaTeX.

5. **Week 5 (10.03-10.09)**
   Complete the Requirements Specification.

6. **Week 6 (10.10-10.16)**
   Starting to code the module to read any kind of XML file and output the content to a JSON file.

7. **Week 7 (10.17-10.23)**
   Adding functions to read the XML formatted file in other file extensions. (for example: PDF, TXT, etc.)

8. **Week 8 (10.24-10.30)**
   The package reads an XML file and extracts certain data of the basic roof to a JSON file and sends it to other areas. Start the Technical specification.

9. **Week 9 (10.31-11.06)**
   Finish writing the Technical specification. Research how to implement security features.

10. **Week 10 (11.07-11.13)**
    Minor bug fixes. Implement some security features.

11. **Week 11 (11.14-11.20)**
    Write some unit tests. Prepare the mid-course presentation. Optimise the module to read more complex roofs measurements.

12. **Week 12 (11.21-11.27)**
    Update our code to read all kind of given roof data. Add more unit tests.

13. **Week 13 (11.28-12.04)**
    Manual testing if the JSON file is identical with the given XML file.

14. **Week 14 (12.05-12.11)**
    Minor bug fixing, testing.

15. **Week 15 (12.12-12.18)**
    Add more unit tests. Bug fixing.

16. **Week 16 (12.19-12.25)**
    Present the final version of our module.

Job distribution:

All the upcoming problems/questions will be discussed with all the team members to find the best possible solution. In this way, everyone will be involved and the overall outcome will be best for the product. Expected workload for each team member is expected to be 20 percent.