VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

SOFTWARE ENGINEERING PROJECT

Technical Specification

Area 5

Done by:

Daria Tovstohan
Olesia Loniuk
Domas Boruta
Sandra Čiuladaitė

Supervisor:
Virgilijus Krinickij
Gediminas Rimša

Vilnius
2022

Contents

1 Introduction

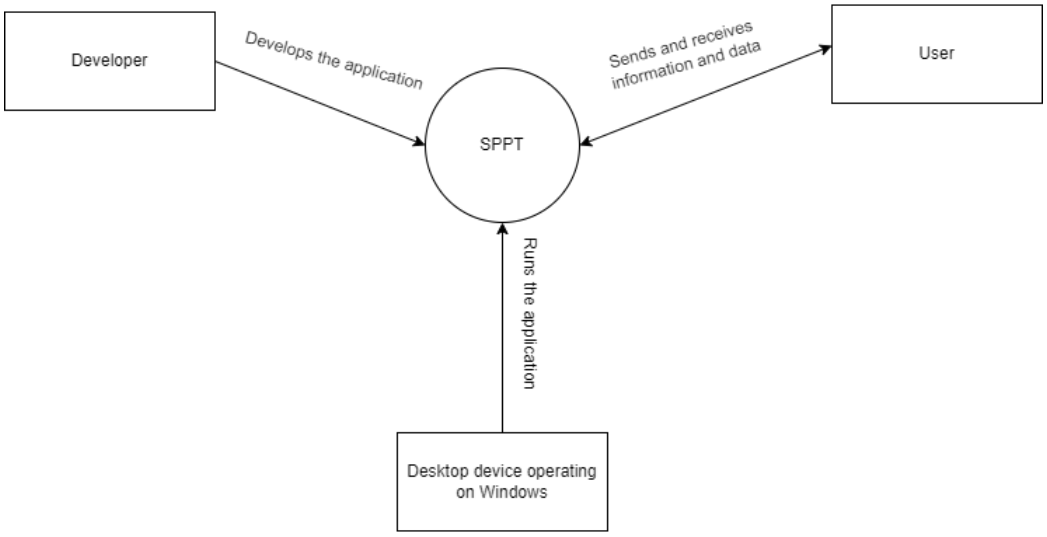
Project name: Integration and visualisation.

Team Distribution:

Team Leader	Daria Tovstohan
Developer	Olesia Loniuk
Developer	Domas Boruta
Developer	Sandra Čiuladaitė

2 Context diagram

An overview of the Solar Panels Placement Tool (SPPT) desktop application will be presented and discussed in this section of the document in order to give a better grasp of how the system will be implemented. To display this data, two different types of diagrams will be used: At the



centre of the diagram, the entire software system is shown as a single process, without any details of the interior structure. The system is surrounded by all of its external entities that interact with the system:

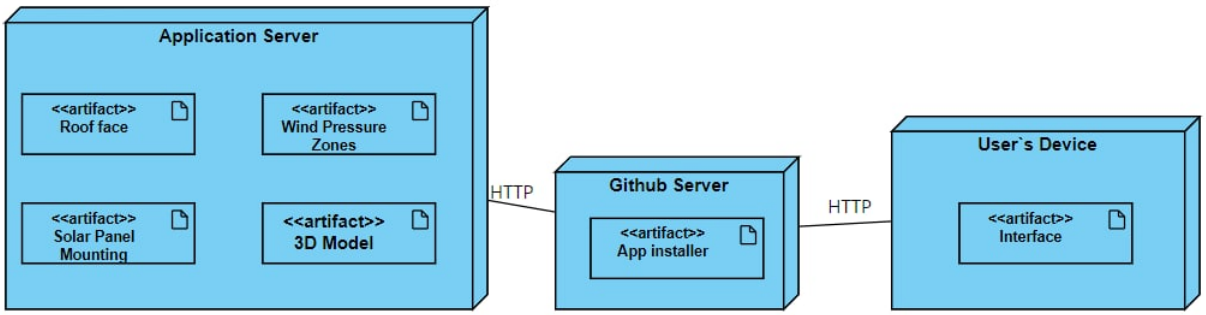
User: The user inputs and then sends data to the application, and in return receive information and data.

Desktop device operating on Windows: The SPPT application is run on a desktop device.

Developer: The developer develops new functionalities to the application, as well as, fixes bugs and issues.

3 UML Deployment diagram

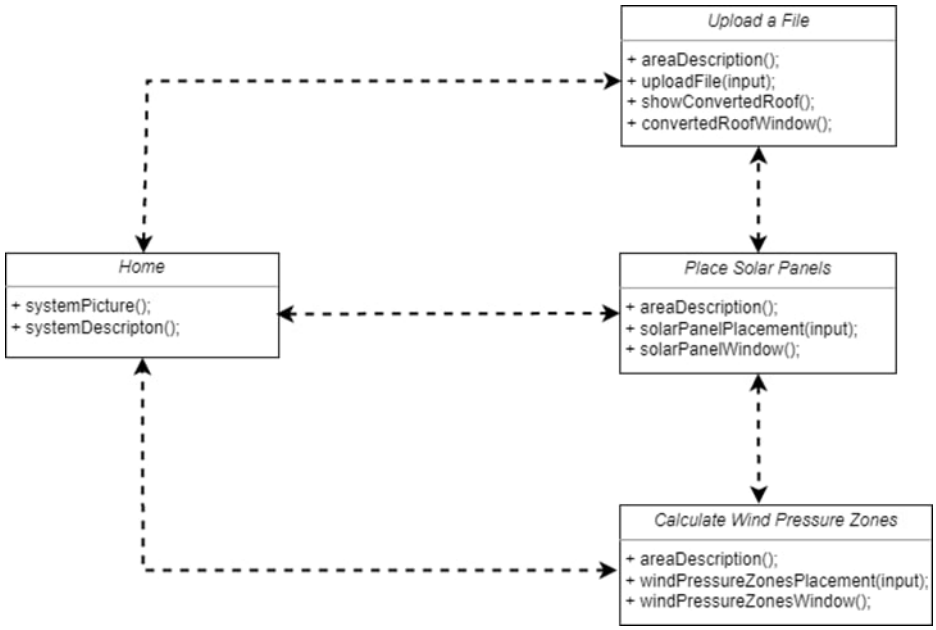
The UML deployment diagram models the physical deployment of software components. In the diagram, hardware components, such as desktop device, is presented as node, with the software components that run inside the hardware components presented as artifacts.



Windows Desktop Application: The user's Windows Desktop Application will be running the SPPT application. The application's code is written in Python programming language, and the UI is formatted using Tkinter.

4 A high-level overview of the system internals

In this section we will explain how the user is going to interact with the system and how it is going to be connected. We will also visualize it with a UML class diagram.



1. After launching the application, the home screen will appear.
2. When at the home screen, the user will see:
    - a picture of the system;
    - a description of the whole system.

The user will also see and be able to select 4 individual tabs at the top of the screen. Those tabs are:

- Home, which is the default tab the user starts with.
- Upload the file.
- Select components.
- Calculate wind pressure zones.

3. 3Upon opening the ‘Upload a File’ tab the user will see the description of Team 1’s work, a file upload box and show converted roof button. After clicking ‘Upload a File’ the user will be able to upload a file with their data. Then, clicking ‘Show converted roof’ will open a window where the user will be able to see a 3D model of the roof dimensions.
4. In the ‘Place Solar Panels’ tab, the user will see the description of Team 3’s work and a button ‘Place’. Upon clicking the button, the user will be able to see a 3D model of the roof with solar panel placement.
5. In the last ‘Calculate Wind Pressure Zones’ tab, the user will see the description of Team 4’s work and a button ‘Calculate Wind Pressure Zones’ which shows a 3D model of the roof with calculated wind pressure zones.

5 Testing

Version control: The Git Hub repository is used weekly for submitting week reports of the work progress(https://github.com/Jamtit/PSI-2022-GR3). **Testing:**

1. **The functionality of the Main Menu buttons**

Preconditions: The desktop application is launched successfully.
Assumption: The user has an internet connection enabled, allowed the application access to the needed information, the user is using a compatible device for running the application.
Test Scenario: The user can get to the area of choice for more information and functionality by clicking on buttons “Upload the file”, “Place Solar Panels”, “Calculate wind pressure zones” or by clicking on the button “Home” from other pages for being redirected back to the main page.
Expected Result: Buttons are redirecting the user to the selected screens.
Steps:

1. The user is clicking on the button of choice.
2. The selected page is viewed.

2. **The functionality of file uploading box**

Preconditions: The desktop application is launched successfully and the user is in the “Upload the file” tab.
Assumption: The user has an internet connection enabled, allowed the application access to the needed information, the user is using a compatible device for running the application.
Test Scenario: The user uploads a file and gets a 3D model of the roof with solar panels.
Steps:

1. The user clicks on the upload button.
2. The user selects the file of the necessary format and uploads it.
3. If the file and its data are in the correct format the 3D model of the roof with solar panels appears.
4. If the file and its data are not in the correct format the error message appears.

6 Technologies and Tools

In this section, the tools and technologies that are used to build the desktop app will be listed and described in detail.

Languages:

- **Python** is going to be the main and only programming language used in the development of the desktop application.

Tools:

- **Visual Studio Code** is a code editor redefined and optimized for building and debugging modern web and cloud applications.
- **Tkinter** is a Python binding to the Tk GUI toolkit.
- **Balsamiq Wireframes** is a graphical user interface website wireframe builder application. It allows the designer to arrange pre-built widgets using a drag-and-drop WYSIWYG editor.

Libraries:

- **PIL:** The PIL adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.
- **Json:** The json library can parse JSON from strings or files. The library parses JSON into a Python dictionary or list. It can also convert Python dictionaries or lists into JSON strings.
- **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.