**Prism Viewer Application - Methodology and Screening Report**

**1. Introduction**

It describes the approach made towards the implementation of the Prism Viewer Application to complete the screening task. The application is providing the facility to the user to view 3D model representations of rectangular prisms and calculate surface area and volume along with visualizing the model through a 3D viewer. It uses both Conda and PIP as a package and has unit testing to validate computations.

**2. Methodology**

**2.1 Application Structure**

The application is developed using Python, primarily leveraging the following tools:

- PyQt5: For creating the user interface.

- pythonocc-core: For 3D visualization of rectangular prisms.

- SQLite: For storing prism dimension data.

- NumPy: For efficient handling of data and calculations.

When a prism is selected from a dropdown, this application retrieves prism dimension values from an SQLite database. Calculating surface area and volume are achieved through the PrismCalculator class, and the 3D model is shown using the 3D rendering engine of OpenCascade.

**3. Unit Tests and Test Strategy**

Unit tests were created to verify the correctness of the surface area and volume calculations for the rectangular prism.

**3.1 Tests Chosen**

The unit tests are implemented in the `test\_prism\_calculator.py` file. The tests cover the following:

- Surface Area Calculation: Tests whether the surface area is calculated correctly using the formula:

Surface Area = 2 x (Length x Width + Width x Height + Length x Height)

- Volume Calculation: Tests whether the volume is calculated correctly using the formula:

Volume = Length x Width x Height

Each test uses sample dimensions for a rectangular prism to check that the output equals the known correct answer. The following tests are self-validating since they compare the actual output of the calculation to the known correct values.

**3.2 Code Example**

Here is an example of the unit test implementation:

import unittest

from prism\_calculator import PrismCalculator

class TestPrismCalculator(unittest.TestCase):

def test\_surface\_area(self):

length, width, height = 3.0, 4.0, 5.0

expected\_surface\_area = 94.0

result = PrismCalculator.surface\_area(length, width, height)

self.assertEqual(result, expected\_surface\_area)

def test\_volume(self):

length, width, height = 3.0, 4.0, 5.0

expected\_volume = 60.0

result = PrismCalculator.volume(length, width, height)

self.assertEqual(result, expected\_volume)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

These tests ensure that the core functionality (area and volume calculation) works as expected.

**4. Conda Packaging Challenges**

**4.1 Packaging with Conda**

The application was packaged using Conda. Below are the major steps taken during the process:

1. Creating a Conda Recipe: The `meta.yaml` file was created to define the application’s dependencies, including Python, PyQt5, NumPy, and `pythonocc-core`.

2. Building the Package: The package was built using `conda-build`, which generated the `.tar.bz2` file.

3. Testing: The Conda package was tested by installing it in a clean environment to ensure that it worked correctly.

**4.2 Challenges Faced**

The main challenges during the packaging process were:

- Dependency Management:  The dependencies, especially for `pythonocc-core`, needed managing and manipulating the Conda Forge channels to ensure the application would not fetch the wrong version of the package.

- Path Issues with Executables: Producing .exe files for the application they wrote using the default method resulted in hard-coded paths to the build environment. Running them in another environment would give them runtime errors. They changed their approach to be using Python gui\_scripts entry point so that they were not to generate .exe files and therefore are allowed to let Python scripts launch the application themselves.

**5. Conclusion**

The Prism Viewer Application is able to complete the tasks as designed, presenting a working graphical interface for computation and visualization of rectangular prisms. Unit tests establish correctness of surface area and volume calculations in addition to packaging the application in Conda and PIP for easy distribution.

**6. References**

- PyQt5 Documentation (https://www.riverbankcomputing.com/static/Docs/PyQt5/)

- pythonocc-core Documentation (http://www.pythonocc.org/)

- Conda Documentation (https://docs.conda.io/projects/conda/en/latest/)