**Implementation Report for Outcomes Report Automator application - Tahmid**

:

**Table of Contents:**

1. **Compilation**

Source files

Directory tree

Programming languages

Compliers

Python libraries used

List of Created Binary Files

1. **Run-time**

list of the COTS

Component Diagram

Deployment Diagram

1. **Testing**

Unit Testing

System Testing

Operational Profile

Non-functional Testing

Tests on application

1. **Traceability**

Design packages vs. Run-time components

1. **Appendix**

Commented Source code

**Compilation:**

For my Outcomes Report Automator application, the source files are in .py format, which is the Python file type. The images used in the application are in .png format, and the application's output is in .pptx format for PowerPoint presentations. The application is developed in Python and utilizes the following libraries: tkinter for the graphical user interface, messagebox for dialog boxes, PIL for image processing, pandas for data manipulation, pptx for PowerPoint generation, and matplotlib for graph plotting. I used PyCharm as my development tool and IDE to create the application. Information on libraries used:

Tkinter - is used for the graphical user interface for creating buttons, labels and check boxes.

Messagebox – is used for the messaged boxes if user enters password correct or wrong to indicate fail or success with a message box.

PIL – this library is used to manipulate the images to create a size which I desire in the application it is used for making a background and used to create a nice size for the company logo

Pandas & matplotlib and used for data manipulation and making the data convert into a graph and then plotting the graph.

pptx -this is used for the PowerPoint allowing the data created with a graph to be outputted into a PowerPoint

**Directory tree diagram of outcomes report Automator application:**

A screenshot of a computer program

Description automatically generated

**List of Created Binary Files**

Main Application Executable:

- File Name: OutcomesReportAutomator.py

Description: This is the main executable file of the application. It is responsible for launching the graphical user interface, handling user inputs, processing data, generating graphs, and exporting the results to a PowerPoint file.

-Location at Run-Time: This file is to be found in the main application directory and which is in the usual directory of the application, usually `C:\Program Files\OutcomesReportAutomator\` or a similar directory where the application is installed.

Supporting Dynamic Link Library (DLL) Files:

File Names: These are currently autogenerated files by the packaging software (such as PyInstaller and cx\_Freeze) and would probably contain the like of tkinter, PIL, pandas, matplotlib, pptx.dll, etc.

Description: These DLL files are the ones essential to the main executable's operation to have the correct functioning of the system. They basically are coding material handling GUI objects, image processing, data manipulation, plot the graph, and PowerPoint generation as well.

Location at Run-Time: They are supposed to be in a special `libs` directory or in some dependency’s subdirectory under the main application directory, e.g., `C:\Program Files\OutcomesReportAutomator\libs\`.

Description of Executables and Their Users

OutcomesReportAutomator.py

- Users This executable will be used by all end-users of the application. They might be analysts, managers, or other stakeholders who need to print out outcomes’ reports. These IT professionals work closely with the end users of the software, who are analysts, managers, and other stakeholders that need it to generate the outcome reports.

-Purpose This executable will be utilized by users to insert data, write diagrams, and develop slideshow presentations. The front-end screen for this application.

**Run-time: Commercial-Off-The-Shelf (COTS) Software Components**

Below are the COTS software components required for the Outcomes Report Automator application:

**Python Interpreter:**

Description: Required to execute Python scripts

File `python.exe`

Location: Typically installed in `C:\Python39\` or a similar directory based on your installation path.

**DLL Files:**

Tkinter DLL:

Description: Required for graphical user interface components.

File: tkinter.dll

Location: Located within the Python installation directory, usually under `Lib\site-packages\tkinter\

PIL DLL (Pillow Library):

Description: Used for image manipulation.

File: PIL.\_imaging.pyd

Location: Located within the Python installation directory, usually under `Lib\site-packages\PIL\`.

Pandas DLL:

Description: Used for data manipulation.

File: pandas. \_libs. Pyd

Location: Located within the Python installation directory, usually under `Lib\site-packages\pandas\\_libs\`.

Matplotlib DLL:

Description: Required for creating charts.

File: matplotlib .pyd

Location: Located within the Python installation directory, usually under `Lib\site-packages\matplotlib\`.

PowerPoint Ribbon DLL:

Description: Used for creating and manipulating PowerPoint files.

File: ribbon.py

Location: Located within the Python installation directory, usually under `Lib\site-packages\native\`.

PIL-Auxiliary DLL:

Description: Used for image format types (e.g., jpg, bmp, png).

File: PIL32.dll

Version: PIL-Auxiliary

python-pptx DLL:

Description: A Python library for working with MS PowerPoint, used for tasks such as reading and writing presentation files.

File: pptx.pyd

Location: Located within the Python installation directory, usually under `Lib\site-packages\pptx\`.

A group of blue rectangular boxes

Description automatically generated**Component diagram:**

A screenshot of a computer

Description automatically generated**Deployment Diagram:**

**Testing:**

A screenshot of a computer program

Description automatically generated**Unit testing:**

|  |
| --- |
| Alternative flow: Incorrect File Generation |
| ID: 1 |
| **Brief description:**  The PowerPoint fails generates the correct report and generates incorrect report for user |
| **Primary actors:**  User |
| **Secondary actors:**  None |
| **Preconditions:**  1. User generates a specific report and clicks Output as PowerPoint |
| **Alternative flow:**   1. The alternative flow starts after step 3 of the normal flow. 2. The application outputs a PowerPoint 3. The PowerPoint information is not correct |
| **Postconditions:**  The python code on the application must have errors or the CSV files which data is coming from must have some errors |

A screenshot of a computer application

Description automatically generated

|  |
| --- |
| Alternative flow: Login fail |
| ID: TM02.2 |
| **Brief description:**  User enters password and is not able to login to the application to use the report generation |
| **Primary actors:**  User |
| **Secondary actors:**  None |
| **Preconditions:**  1. User starts the application and then try to login |
| **Alternative flow:**   1. The alternative flow starts after step 1 of the normal flow. 2. User enters password and gets error message” Invalid password” |
| **Postconditions:**  User has either forgot password or developers have programmed the application with incorrect password |

A screenshot of a report

Description automatically generated

|  |
| --- |
| Alternative flow: Incorrect Report Outputted |
| ID: TM02.2 |
| **Brief description:**  User generates SDQ baseline report, but demographics report is generated instead. |
| **Primary actors:**  User |
| **Secondary actors:**  None |
| **Preconditions:**   1. User selects SQD baseline report out of the 3 options 2. Then clicks generate as PowerPoint |
| **Alternative flow:**   1. The alternative flow starts after step 4 of the normal flow. 2. Application displays a message: “PowerPoint report generated successfully” 3. User opens PowerPoint and sees that incorrect report has been generated |
| **Postconditions:**  Developers are called as this is an error in the application and they fix the code. |

**System Testing:**

Expected Operational Profile

1. User Login:
   * Frequency: 4 times per day per user.
   * Justification: Users access the application multiple times daily.
2. Report Selection:
   * Frequency: 10-15 times per week.
   * Justification: Regular report generation is a core function.
3. Data Upload/Processing:
   * Frequency: 5-10 times per week.
   * Justification: Regular data updates require frequent processing.
4. Graph Generation:
   * Frequency: 5-10 times per week.
   * Justification: Frequent need for visual data analysis.
5. PowerPoint Report Creation:
   * Frequency: 3-5 times per week.
   * Justification: Reports are created for presentations and meetings.
6. Exit Application:
   * Frequency: 4 times per day per user.
   * Justification: Matches login frequency.

**Automated System Testing**

1. Automation Framework:
   * Use tools like Selenium, JMeter, and Python scripts.
2. Populating Parameters:
   * Automate form interactions and test case development for each use case.
3. Judging Results:
   * Implement assertions and validation mechanisms to compare actual results with expected outcomes.
4. Recording Testing Results:
   * Use logging, automated test reports, and structured data storage for analysis.

**Operational Profile:**

**Expected Operational Profile with Probability Distribution**

User Login: Probability = 0.291

Report Selection: Probability = 0.165

Data Upload/Processing: Probability = 0.099

Graph Generation: Probability = 0.099

PowerPoint Report Creation: Probability = 0.053

Exit Application: Probability = 0.291

The sum of all probabilities is 1.0.

**Automated System Testing**

Automation Framework

Selenium for GUI interaction, JMeter for performance testing, and Python scripts for data validation.

**Populating Parameters**

Use Selenium to automate form interactions and submit test data.

Example use cases: Automate login, report selection, data upload, graph generation, and PowerPoint creation.

**Judging Results**

Implement assertions to compare actual results with expected outcomes.

Validate successful logins, correct report generation, data processing accuracy, proper graph visualization, and PowerPoint content.

**Recording Testing Results**

Use logging within test scripts for detailed run information.

Generate automated test reports using tools like Allure or HTMLTestRunner.

Store results in a structured format for analysis.

**Non-functional Testing:**

A screenshot of a computer error

Description automatically generatedA close-up of a text

Description automatically generated**Security:**

**Scailabity:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated**Reliability and Availability:**

A white text with black text

Description automatically generated

A black text on a white background

Description automatically generated

Tests on Outcomes report Automator application:

Test performed to see if entry box is left blank then error message should appear test successful:

A screenshot of a computer screen

Description automatically generated

Test to see if user types wrong password or username on entry box then error message appears **successful** A screenshot of a computer screen

Description automatically generated

**Test to see if user enter correct password, then shows user is successful and allows user to go to the next page: successful**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Test to make sure after use selects Output as report PowerPoint is generated successful.**

A screenshot of a computer

Description automatically generated

A blurry image of a city

Description automatically generated

A screenshot of a computer

Description automatically generated

**Traceability:**

A black screen shot of a program

Description automatically generated

**Appendix:**

**Example of all the different library imported for the application:**

A screenshot of a computer

Description automatically generated

**Example of where the images library PIL was used:**

A screenshot of a computer program

Description automatically generated

**Example of where messagebox was used in the login page:**

A screenshot of a computer screen

Description automatically generated

**Example of where the GUI was used for the report selection:**

A computer code on a black background

Description automatically generated

A computer screen shot of a program

Description automatically generated**Example of where pandas, matplotlib and pptx are used to create the PowerPoint**A screenshot of a computer program

Description automatically generated