Master Program Documentation

Introduction:

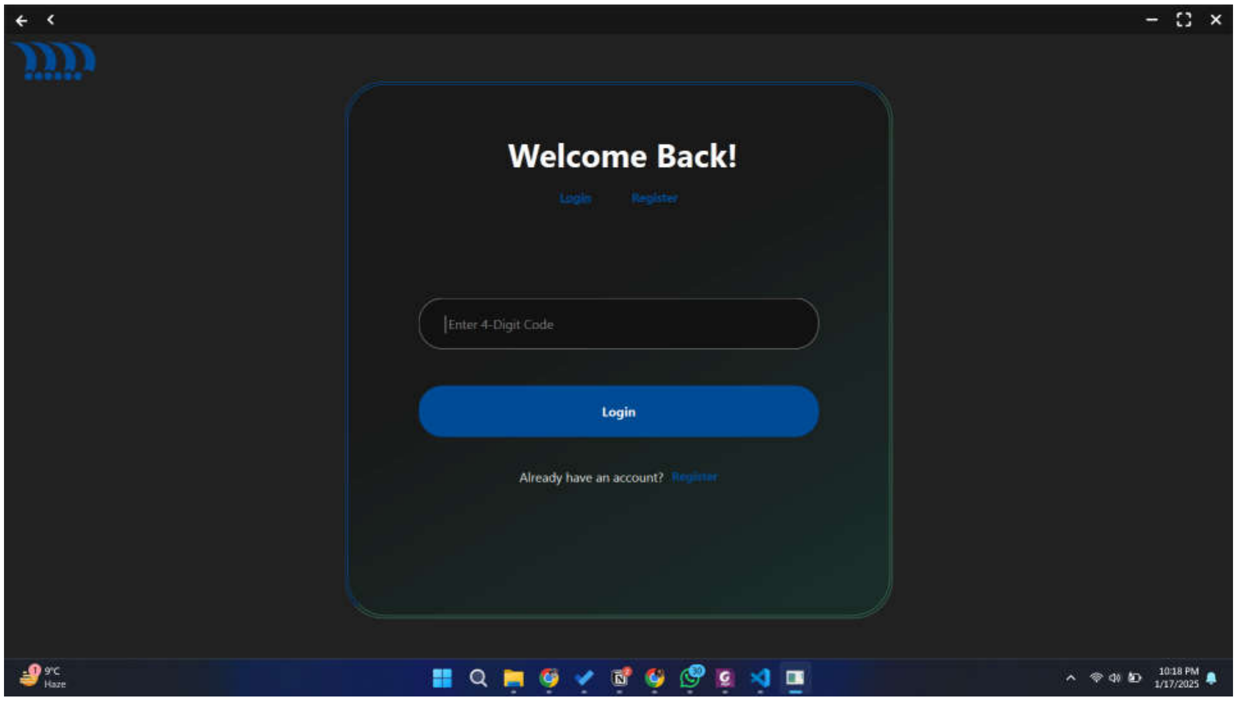
- **Introduction:** This Master Program seamlessly integrates three independent projects — SMT, Fire test, and QR Code Generator — into a unified, user-friendly platform. This program is designed to simplify workflows and enhance efficiency by allowing users to manage all three programs from a single interface.

Installation:

- Run Command "pip install -r requirements.txt" to install required libraries.
- Install gtk3-runtime
- Run the application by running the command "python main.py".

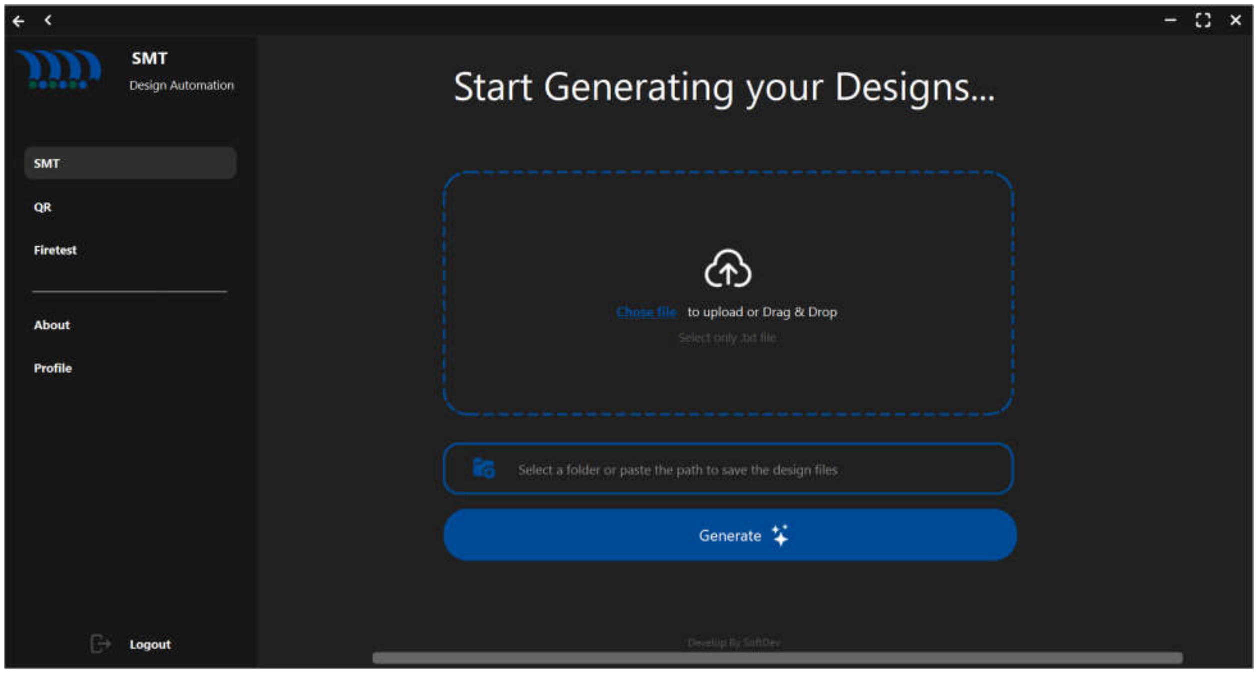
Usage:

Register a user or login via 4 digit pin.

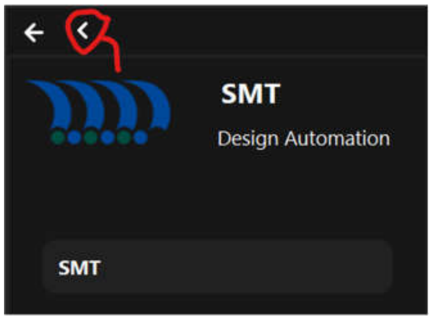


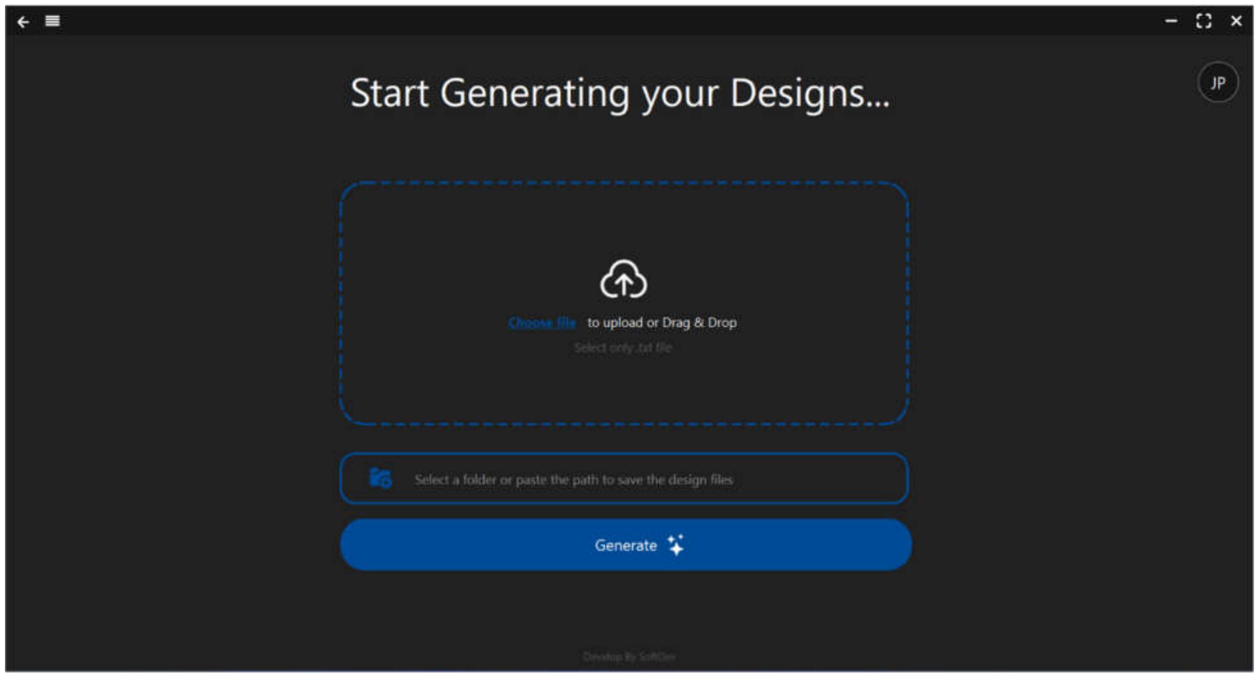
you can press “enter” for major action buttons like login, generate etc

How to use SMT module:


SMT
Design Automation
SMT
QR
Firetest
About
Profile
Logout
Start Generating your Designs...
Chose file to upload or Drag & Drop
Select only .txt file
Select a folder or paste the path to save the design files
Generate
Develop By SoftDev

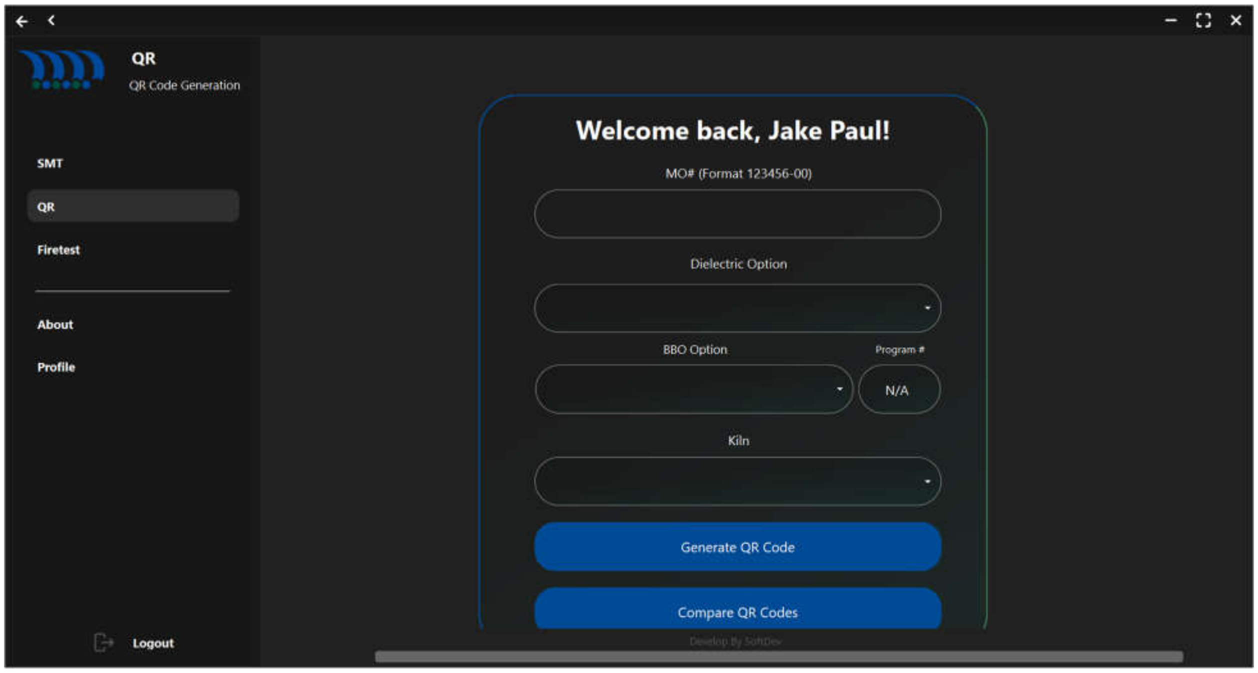
side window can be hidden by pressing this




JP
Start Generating your Designs...
Choose file to upload or Drag & Drop
Select only .txt file
Select a folder or paste the path to save the design files
Generate
Develop By SoftDev

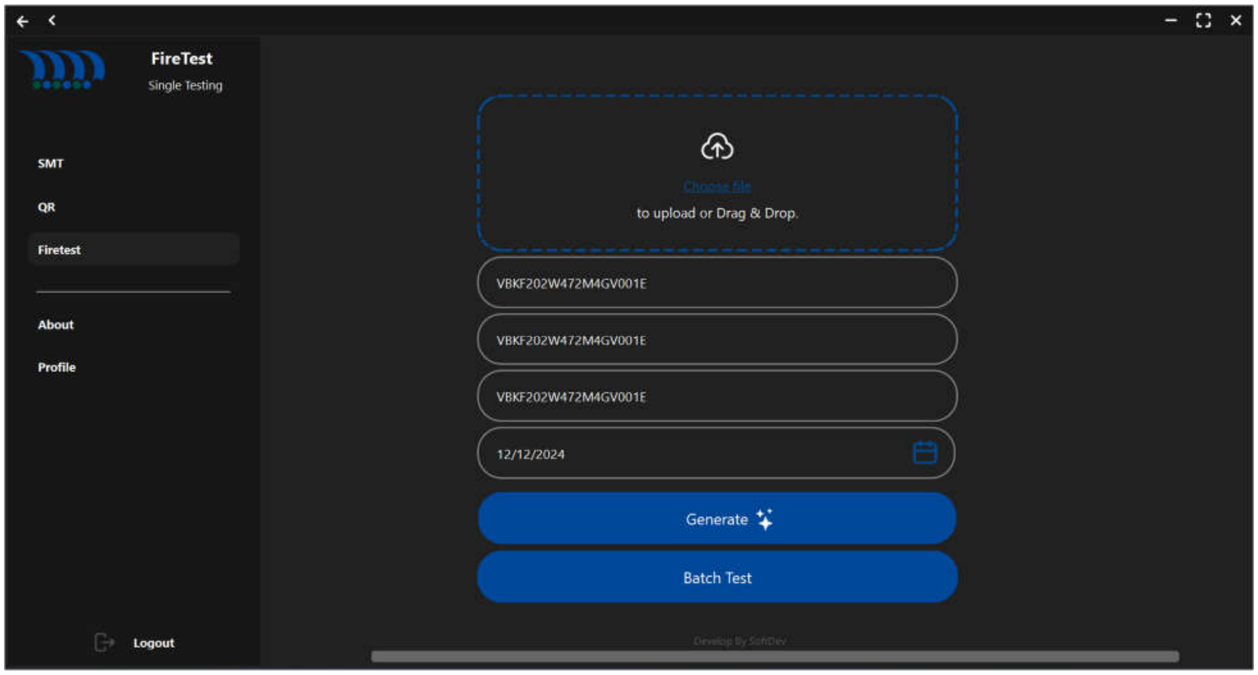
- Generate design by choosing txt file containing part numbers and output folder for SMT design.
- Enter required values by following instructions in prompt and continuing.
- Review files and create build sheet. Build sheet pdf and prn files generated.
    - Excel needs to be installed on the PC where you will run the Python file. The Python script exports the build sheet in both PDF and PRN formats to the output folder. It also updates the NCCDB.xlsx database, which should be in the same directory as the NCC Layer Build Sheet Maker.xlsm file.
- View buildsheet and share for approval.
- Enter email to share files.

How to use QR module:



- Enter required input for generating QR.
- QR is available in pdf in QR CODE PDFs folder.
- Data entered in generating QR saved in qr_data.xlxs.
- Compare QR codes by entering operator and traveler data in their respective fields.
- In case of match, data updated in excel.

How to use Firetest module:



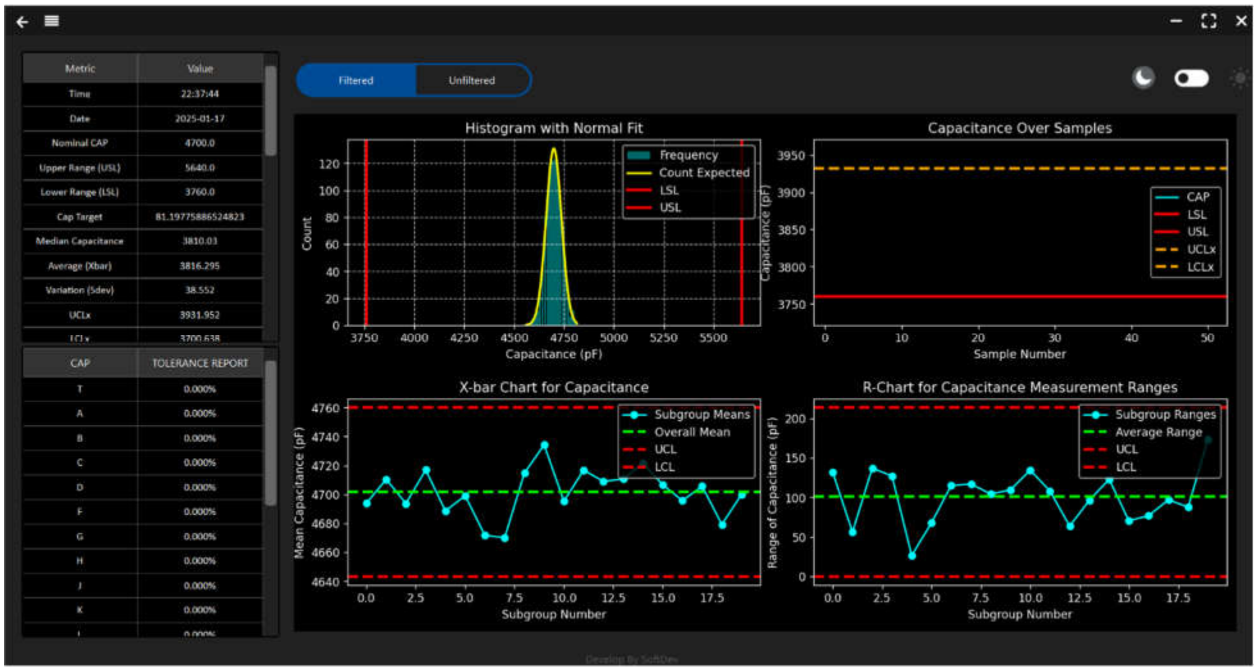
VBKF202W472M4GV001E

VBKF202W472M4GV001E

VBKF202W472M4GV001E

12/12/2024

- Firetest in case of a single file, upload the data excel sheet and required inputs.
- Graphs available in both dark and white themes.
- Output Table and tolerance yield table.
- Filtered and unfiltered data options.
- Data stored in an Excel Database.
- In case of batch testing, choose input and output folder. Batch testing—outputs are available in HTML format for both filtered and unfiltered data options with black and white themes.



Metric	Value
Time	22:37:44
Date	2025-01-17
Nominal CAP	4700.0
Upper Range (USL)	5640.0
Lower Range (LSL)	3760.0
Cap Target	81.19775886524823
Median Capacitance	3810.03
Average (Xbar)	3816.295
Variation (Sdev)	38.552
UCLx	3931.952
LCLx	3700.638

CAP	TOLERANCE REPORT
T	0.000%
A	0.000%
B	0.000%
C	0.000%
D	0.000%
F	0.000%
G	0.000%
H	0.000%
J	0.000%
K	0.000%
L	0.000%

Project Structure:

- **README.md:** Description of the project.
- **requirements.txt:** List of Python dependencies required for the project.
- **main.py:** Entry point of the application.
- **ui/:** Directory containing UI-related files.
    - **main_window.ui:** Main window layout file. i.e interface.ui
- **src/:** Source code directory.
    - **SMT/:** SMT Project Directory
        - **Buildsheet_Automation/:**
            - **Outputs/:** In this folder output of Buildsheet_Automation is saved in two files. One is pdf and second is .prn file
            - **buildsheet_automation.py:** This script automates the process of updating an Excel file, running a macro, and organizing the output files
            - **NCCDB.xlsx:** This is database used in SMT.
            - **NCC LAYER BUILD SHEET MAKER.xlsm:** This file was used in buildsheet_automation.py.
        - **Design_Automation/:**
            - **assets/:** contain multiple .xlsx files that are used in SMT
            - **src/:**
                - **design.py:** This script is responsible for retrieving design and dielectric thickness data based on a given part number.
                - **dielectric.py:** This script focuses on retrieving dielectric information for a given part number based on its subfamily type and dielectric value.
                - **layers.py:** This code involves calculations related to the design and manufacturing of components, such as calculating capacitance, active area, and dielectric thickness. It includes back-calculation methods to adjust parameters like margin, dielectric thickness, and the number of layers based on required capacitance and design constraints.
                - **part_information.py:** This script is designed to handle capacitor part numbers and retrieve legacy values and

dimensions based on a dataset provided in an Excel file. It utilizes a detailed structure to classify subfamilies and filter dimensions from the dataset, with provisions to interact with the user for choices when multiple options exist. The get_legacy_value_and_part_dimensions function, in particular, decodes the part number into subfamily, size, and voltage, then proceeds to retrieve the corresponding legacy value and dimensions.

      - **text_file_processing.py:** This script is designed to extract part numbers and corresponding MO numbers from a text file.
      - **translation.py:** This code provides functions to translate between legacy part numbers (LPN) and global part numbers (GPN) for electronic components, using mappings defined in an Excel file

    - **design_automation.py:** The script automates the process of calculating and updating design parameters based on given part numbers and other relevant component information. It works with Excel files to update the necessary design values and handles different dielectric materials and capacitor designs based on input from a text file.

  - **smt_handler.py:** The SMTHandler class is a central part of a user interface designed for handling SMT (Surface Mount Technology) processes. This class integrates design automation and buildsheet generation in an interactive application, allowing the user to upload files, generate designs, and generate build sheets.

- **QR/:** QR Project Directory
  - **assets/:** This contains three txt files i.e. bbo_options.txt, dielectric_options.txt and kiln_options.txt. These files contain configuration options for respective files. This also contains a Json file that is used in qr_handler.py.
  - **qr_data.xlsx:** This data file is used in QR code generation
  - **qr_handler.py:** It manages the generation, comparison, and updating of QR codes, along with interaction with external data files (Excel, text files, and JSON) to support the application's functionality.

- **Firetest/:** Firetest Project Directory
  - **assets/:** The Mapping.xlxs file is used in generating tables and different charts in Firetest.

      - **src/:**
        - **batch_testing.py:** The overall script consists of eight batch testing functions, each designed to process data from a directory structure and generate HTML reports with results for various testing types. These functions focus on handling different conditions (filtered and unfiltered, dark and white), while also managing errors and logging issues.
        - **firetest_Final_Dark.py:** The Filtered_dark_function, Unfiltered_dark_function, and batch_test collectively form a data processing pipeline designed to analyze, visualize in dark themes, and report data for a batch of test files associated with different part numbers.
        - **firetest_Final_White.py:** The Filtered_dark_function, Unfiltered_dark_function, and batch_test collectively form a data processing pipeline designed to analyze, visualize in white theme, and report data for a batch of test files associated with different part numbers.
        - **yield_of_tolerance.py:** This script is designed for calculating the yield percentage of a product based on its tolerance range and the data provided in an Excel sheet. The code has several core functions, which work together to process the part number, fetch tolerance data, and compute the results.
      - **data.xlsx:** This file is selected as input in Firetest.
      - **Database.xlsx:** Database file that saves result of Firtest.
      - **firetest_handler.py:** This script handles data generation for both filtered and unfiltered fire tests, calculates yield results, and displays them in tables. Users can switch between dark and white themes for the UI and plots. The code also includes functionality for batch testing, allowing users to process multiple files simultaneously and generate reports.
  - **messagebox.py:** Additional helper function for displaying message boxes in APP.
  - **shortcut_handler.py:** Additional helper function for implementing shortcuts.
  - **ui_main_window.py:** Automatically generated Python code from main_window.ui. i.e. ui_interface.py
- **qss/:** Directory for Qt Style Sheets (QSS) and icons.
  - **scss/:** SCSS files for styling.
  - **icons/:** Icon images.
- **logs/:** Directory for log files.

    - **custom_widgets.log:** Log file.
- **json_styles/:** Directory for JSON style files.
- **style.json:** Example JSON style file.
- **generated-files/:** Directory for files auto-generated by the custom widget's module.
    - Example generated files include **UI's** and **JSon** files
- **assets:/** Directory for user-provided assets, such as images, icons, or other resources. **icons:/** Icon images. **Lato:/** Font file. **user_data.json:** Contains registration information of users.
- **SMT Output Folder:/** Directory contains Design Files generated from SMT Project.
- **QR CODE PDFs:/** PDFs containing QR that are generated from QR Project.
- **text_file.TXT:** This file is selected in the SMT module by clicking "Choose File." It contains a list of PART NUMBERS, which are essentially orders for which the designs will be generated in the SMT application. The file can have alternate text, but type remains the same i.e. ".txt".

This structure allows for automatic conversion of UI files to Python code and placement within the src folder, simplifying the development process for users.

----------------------------*End Of Documentation*--------------------------------