**1. Introduction**

The proposed Excel automation tool aims to streamline data extraction and conversion by seamlessly transforming user-input data into standardized formats, represented by ramp up planning table and VRFC table.

**2. Project Description**

1. **User Input Fields**: (to be discussed)
   1. Time scale (year, quarter, week)
   2. Number of testers
   3. Percentage of
   4. Testing and transiting time (in weeks)
2. **Output Format**: Standardized formats, represented by ramp up planning table and VRFC table.
3. **Automation Process**:

Normal Flow:

* 1. User select the table they want to convert in the tools’ user interface
  2. User input required fields
  3. User specify the file location and file name
  4. User clicks “generate” and the tool will convert the info
  5. User get the output excel files

Validation and Formatting: (to be discussed)

The tool will validate the input data to ensure accuracy and consistency. It will also apply predefined formatting rules to maintain standardization across 2 tables.

1. **Key Features**:
   1. Reduce repetitive works from the user to key in, copy and paste

**3. Technical Approach**

Describe the technical aspects of the project, including:

* **Python Libraries**: openpyxl, pandas, and GUI library
* **User Interface**:

A screenshot of a computer

Description automatically generated

* **Error Handling**: The tool will check the input fields to ensure the inputs are valid.

**4. Deliverables**

* **Executable Tool**: The fully functional Excel automation tool.
* **Documentation**: User manual or guide explaining how to use the tool.
* **Source Code**: Provide access to the source code for future maintenance and enhancements.

Modules:

a. User Interface Module:

* Tkinter, PyQt, or wxPython.
* Implement user input forms and controls for users to input data or specify parameters for Excel sheet generation.

b. Data Processing Module:

* Define functions or classes responsible for processing user-input data.
* Implement data validation, transformation, and formatting logic based on the requirements for generating Excel sheets.

c. Excel Generation Module:

* Utilize Python libraries like **openpyxl**, **pandas**, or **xlsxwriter** for creating and manipulating Excel files.
* Define functions or classes to dynamically generate Excel sheets based on processed data.

d. Error Handling Module:

* Implement error handling mechanisms to detect and handle exceptions gracefully.
* Provide informative error messages to guide users in resolving issues encountered during data processing or Excel generation.