# Module Interface Specification
# **Formulate**

Team 25, MECHTRON 4TB6
Ahmed Nazir, nazira1
Stephen Oh, ohs9
Muhanad Sada, sadam
Tioluwalayomi Babayeju, babayejt

April 5, 2023

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| 2023/01/18 | 1.0 | Final Version |
| 2023/04/05 | 2.0 | Revision 1 |

# 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/ahmed-nazir/Capstone/blob/main/docs/SRS/SRS.pdf

# Contents

# 3 Introduction

The following document details the Module Interface Specifications for the Formulate system. Formulate enables teams to streamline data collection and storage, resulting in testing overhead reduction and increased control of raw test data gathered by automating aspects of the testing procedure.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/ahmed-nazir/Capstone.

# 4 MIS

This section describes all the modules and their respective classes and/or functions. It should be noted that exception handling only applies to some modules as some of them are automatically generated by external software and do not require error handling.

## 4.1 Module - ui_main.py

### 4.1.1 Description

Python file generated by PyQt designer which sets up the application's window and its design

### 4.1.2 Classes

**Class:** Ui_MainWindow() - Contains all methods for setting up the application's window and its static front end design

| Methods | Parameters | Return |
|---|---|---|
| setupUi() - Takes a PyQt MainWindow object and sets up it's layout according to the ui file created in designer | Self, MainWindow [QMainWindow] | None |
| retranslateUi() - Sets the static text of the GUI's buttons and labels | Self, MainWindow [QMainWindow] | None |

## 4.2 Module - ui_functions.py

### 4.2.1 Description

Imports all necessary libraries for backend functions, creates connection to database, and contains class for UI functions

### 4.2.2 Classes

**Class:** UIFunctions() - Contains the functions that are connected to buttons in the application's UI

| Methods | Parameters | Return |
|---|---|---|
| toggleMenu() - Handles the animation for toggling the side menu | Self, maxWidth [integer], enable [boolean] | None |
| login_into_app() - Checks if the enter username/password are valid and correct and signs user into their account | Self | None |
| continue_signup() - Checks if all the sign up fields are valid and stores account/login details in database | Self | None |
| move_to_submit_test() - Ensures that only logged in users are allow to reach the submit test page | Self | None |
| browse_and_display_pictures() - Opens a popup where users are allowed to navigate through their file directory, select a picture, and display that picture in the UI | Self | None |
| upload_test_info() - Submits test data and user-inputed test information to the database, and uploads test picture to azure blob storage | Self | None |
| startTest() - Starts the test on the device and begins collecting data | Self | None |
| runProg() - Creates another thread which will allow the GUI to be operable while it is conducting tests | Self | None |
| stopTest() - Stops reading values from the Arduino and gathers all the data in a viewable table | Self | None |
| declineData() - Erases the data collected from the last test and does not submit it to the database | Self | None |
| submitData() - Submits data from the test to the Azure database | Self | None |
| retrieveData() - Retrieves data from the local onboard storage | Self | None |
| connect_wireless() - Connects to the device via Wi-Fi | Self | None |
| disconnect_wireless() - Disconnects the PC from the device Wi-Fi network | Self | None |
| connectWired() - Connects to the device via COM port (wired connection) | Self | None |
| disconnect_wired() - Disconnects from the COM port currently connect | Self | None |
| view_dashboard() - Directs user to PowerBi dashboard on their browser | Self | None |
| ping() - Retrieves the currently configured sensors and displays it for the user | Self | None |
| select_com() - Checks the current COM ports detected and displays them in a drop down | Self | None |

| Methods | Parameters | Return |
|---|---|---|
| make_config_page() - Loads the saved sensors in the configuration page drop down | Self | None |
| autofill_config() - Automatically fills out the fields in the sensor config menu when users select a saved sensor | Self | None |
| gen_config_sensors() - Generates the Arduino code if the fields are not empty | Self | None |
| saveConfiguration1() - Saves the sensor configuration for sensor 1 in savedSensors.json | Self | None |
| saveConfiguration2() - Saves the sensor configuration for sensor 2 in savedSensors.json | Self | None |
| saveConfiguration3() - Saves the sensor configuration for sensor 3 in savedSensors.json | Self | None |
| saveConfiguration4() - Saves the sensor configuration for sensor 4 in savedSensors.json | Self | None |

### 4.2.3 Functions

| Function | Parameters | Return |
|---|---|---|
| hash_new_password() - Generates a hashed password based on the user's inputted password | password [string] | salt [string], hashed_pass [string] |
| is_correct_password() - Checks if inputted password matches stored password in database | salt_hex [string], stored_hash [string], pass_to_check [string] | Boolean |
| connect_to_database() - Creates a connection to the database to be able to read and write values from the application to the database | None | conn [connection object], cursor [cursor object] |

### 4.2.4 Exception Handling

Input validation of the user information and try/except statements are the main forms of exception handling. User fields for signing up are checked to ensure that they are not empty and that the password follows the rules of having 8 minimum characters and includes an alphabet, number, and an non-alphanumeric character. When logging in, inputted passwords are checked to ensure that they match the passwords stored in the database. In addition, most functions preform their work in a try statement and deal with errors in a except statement. Connectivity to the device and logged in state of users are checked before they are able to move to 'Start Test' and 'Submit Test' pages, respectively. Furthermore, checks for empty user fields or test information are checked before test data is submitted. If any of these conditions fail, an exception is raised which gets handled in the except statements. The except statements create an error popup with the appropriate message to signal to the user where they have gone wrong.

## 4.3 Module - arduino_code_generator.py

### 4.3.1 Description

This module generates Arduino code based on the users inputs in the configure sensors page. It produces a mainArduino.ino file which the user can use to flash their Arduino.

### 4.3.2 Classes

**Class:** CodeGenerator() - Contains the functions that write the Arduino code

| Methods | Parameters | Return |
|---|---|---|
| generate() - gets the data which give information on the sensors attached and generates the mainArduino.ino file | data | None |

## 4.4 Module - main.py

### 4.4.1 Description

Imports backend functions and frontend setup of GUI. This is also used to start and run the desktop application

### 4.4.2 Classes

**Class:** MainWindow() - Initializes a PyQt main window that is defined in ui_main.py and connects the buttons in the desktop application's UI to backend functions defined in ui_functions.py

| Methods | Parameters | Return |
|---|---|---|
| __init__() - Initializes the application and connects UI buttons to backend functions | Self | None |
| changeText() - Add text to menu buttons when toggling full side menu and vice-versa | Self | None |

## 4.5 Module - resource_rc.py

### 4.5.1 Description

Python file generated by PyQt resource compiler and sets up all the PyQt resources (local images) to be displayed during runtime of application

### 4.5.2 Functions

| Function | Parameters | Return |
|---|---|---|
| qInitResources() - Registers the raw byte data of each image to the Qt resource system | None | None |
| qCleanupResources() - Unregisters the raw byte data of each image to the Qt resource system | None | None |

## 4.6 Module - mainArduino.ino

### 4.6.1 Description

This module runs on the Arduino and collects all the data from the various sensors connected to it. It also takes the data and sends it to the PC wired or wirelessly.

### 4.6.2 Functions

| Function | Parameters | Return |
|---|---|---|
| setup() - Initializes all the sensors, SD card module and the serial communication lines between the PC and Wi-Fi module | None | None |
| loop() - This function reads data from the sensors and creates a bytestring to send to the PC | None | None |

## 4.7 Module - mainESP8266.ino

### 4.7.1 Description

This module runs on the NodeMCU (ESP8266) and allows for the Arduino to send data to it and relay that information to the PC via Wi-Fi.

### 4.7.2 Functions

| Function | Parameters | Return |
|---|---|---|
| setup() - Initializes the ESP8266 as a wireless access point so our PC can connect to it, it also initializes the serial port to allow for communication between the PC and Arduino | None | None |
| loop() - This function acts as a relay to pass information sent from the Arduino to the PC via TCP and also send information from the PC to the Arduino | None | None |