Sheet Metal Client Hub Test Case Document

Document Title: Test Case Document
Date: 28 May 2025
Prepared by: Laurie Moffat
Course: PDSWD7 PDA in Software Development Level 7, Fife College, Semester 1, 2024/25

Introduction

This Test Case Document details the test cases for the Sheet Metal Client Hub application, as part of the Testing phase (May 21–31, 2025) in the Waterfall SDLC, following the Test Plan (#5_Test_Plan.pdf). The application is a Tkinter-based GUI for sheet metal part cost calculation and quote generation, with file I/O for user credentials, rates, and outputs. The test cases validate the GUI functionality, cost calculation logic, secure file I/O, and unit tests for core components, ensuring compliance with functional requirements and design specifications.

Testing Objectives

1) Verify GUI functionality across login, part input, cost output, and settings screens.
2) Validate cost calculations for 10 work centres (Cutting, Bending, Welding, Assembly, Finishing, Drilling, Punching, Grinding, Coating, Inspection) across all material thicknesses (1, 1.2, 1.5, 2, 2.5, 3 mm), lay-flat dimensions, and bends etc.
3) Ensure secure file I/O for user credentials (`users.json`), rates (`rates.json`), output (`output.txt`), quotes (`quotes.txt`), and parts catalogue (`parts_catalogue.txt`).
4) Confirm unit tests cover core functionality in `calculator.py`, `gui.py`, and `logic.py`.

Test Scope

- ☑ In-Scope: Functional testing of GUI and calculations, unit testing of code, file I/O validation.
- ☒ Out-of-Scope: Performance testing, security penetration testing.

Test Environment

- ❖ Location: `C:/Sheet-Metal-Client-Hub`
- ❖ Python Version: 3.13.2
- ❖ Dependences: Tkinter, PIL
- ❖ Files: `src/gui.py`, `src/utils.py`, `src/main.py`, `src/logic.py`, `src/file_handler.py`, `src/calculator.py`, `src/logger.py`, `src/logging_config.py`, `data/users.json`, `data/rates.json`, `data/output.txt`, `data/quotes.txt`, `data/parts_catalogue.txt`, `docs/images/laser_gear.png`
- ❖ Logs: `data/log/gui.log`, `data/log/logic.log`, `data/log/utils.log`

GUI Functionality Test Cases				
Test ID	**Description**	**Preconditions**	**Steps**	**Expected Result**
TC-GUI-01	Verify login with valid User credentials	`users.json` contains `laurie` (p	1. Run `python src/main.py`. 2. E	Part input screen loads with `#f0f0f0` he
TC-GUI-02	Verify login with valid Admin credentials	`users.json` contains `admin` (p	1. Run `python src/main.py`. 2. E	Admin settings screen loads, `#28a745`
TC-GUI-03	Verify login with invalid credentials	`users.json` as above	1. Enter username `laurie`, passwor	Error message: “Invalid username or pas
TC-GUI-04	Verify part input screen for Single Part	Logged in as `laurie`, `parts_cata	1. Select Single Part tab. 2. Enter `	Part added to listbox (`PART-123 (10)`),
TC-GUI-05	Verify part input screen reset	Logged in as `laurie`, part added	1. Add part as in TC-GUI-04. 2. Clic	Inputs reset (Part ID: `ASSY-`, material: `
TC-GUI-06	Verify part search pop-up	Logged in as `laurie`, `output.txt`	1. Click “Add Part to List”. 2. Searc	Part added to listbox (`PART-123 (5)`), s
TC-GUI-07	Verify quote screen functionality	Logged in as `laurie`, parts adde	1. Click “Quote”. 2. Enter custome	Quote generated, total calculated based
TC-GUI-08	Verify admin settings screen	Logged in as `admin`	1. Create user `testuser`, password `	User created/removed in `users.json`, r
TC-GUI-09	Verify admin credentials dialog	Logged in as `laurie`	1. Click “Settings”. 2. Enter `admi	Admin settings screen loads, log shows
TC-GUI-10	Verify clear parts list	Logged in as `laurie`, multiple pa	1. Add parts via TC-GUI-04, TC-GUI-06	Listbox cleared once, `added_parts` em

Cost Calculation Test Cases				
Test ID	Description	Preconditions	Steps	Expected Result
TC-COST-01	Verify cost for Mild Steel, 1.0mm, Cutting	Logged in as `laurie`, `rates.json	1. Single Part, `PART-101`, revision `A	Cost calculated (material + Cutting cost,
TC-COST-02	Verify cost for Stainless Steel, 2.5mm, Be	As above	1. Single Part, `PART-102`, revision `B	Cost calculated (material + Bending cost
TC-COST-03	Verify cost for all material thicknesses	As above	1. For each thickness (1, 1.2, 1.5, 2, 2.	Costs vary by thickness (higher for thicke
TC-COST-04	Verify cost with all 10 work centres	As above	1. Single Part, `PART-109`, revision `A	Cost includes all operations, added to li
TC-COST-05	Verify cost with fasteners	As above, `parts_catalogue.txt` va	1. Single Part, `PART-110`, revision `A	Cost includes £500 (50 * £10.0), added to

File I/O Test Cases				
Test ID	**Description**	**Preconditions**	**Steps**	**Expected Result**
TC-FIO-01	Verify user credentials read	`users.json` has `laurie` (hash:	1. Login with `laurie:moffat123`.	Login succeeds, log shows `Credentials
TC-FIO-02	Verify rates read and write	`rates.json` has valid rates	1. Login as `admin`. 2. Update `n	Rate updated in `rates.json`, log shows
TC-FIO-03	Verify output file save	`output.txt` writable	1. Calculate cost for `PART-123` (TC-G	Part details saved (e.g., `PART-123,A,Mil
TC-FIO-04	Verify parts catalogue read	`parts_catalogue.txt` has `FAS-00	1. Add fastener `FAS-001` in Single Pa	Cost includes £500 (50 * £10.0), log show
TC-FIO-05	Verify quote file save	`quotes.txt` writable	1. Generate quote (TC-GUI-07). 2. (	Quote saved with customer `FatMan`, to

Unit Test Cases				
Test ID	**Description**	**Preconditions**	**Steps**	**Expected Result**
TC-UNIT-01	Test calculator cost function	`calculator.py` accessible, mock	1. Run unit test for `calculate_cost` w	Returns correct cost (e.g., £753 based on
TC-UNIT-02	Test GUI login validation	`gui.py` accessible, mock `file_h	1. Run unit test for `login` with mock	Returns “Login successful as User”, test
TC-UNIT-03	Test logic save function	`logic.py` accessible, mock `file_	1. Run unit test for `calculate_and_sa	Saves to mock `output.txt`, returns total
TC-UNIT-04	Test utils hash function	`utils.py` accessible	1. Run unit test for `hash_password`	Returns hash `4b5a1911...`, test passes