



Kwantlen Polytechnic University

School of Business

Final Report

Financial CRM System

Integration Project II
INFO 4290 S10

Group members:

Harman Dhaliwal
100384897
harman.dhaliwal5@email.kpu.ca

Jas Mangat
100317501
jaskaran.mangat@email.kpu.ca

Mansimar Kaur
100345733
mansimar.kaur@email.kpu.ca

Neil Maedel
100315152
neil.maedel@email.kpu.ca

Submitted on:
08/14/2022

Table of Contents

Purpose of the Document	1
Project Background	1
Technology and Market Analysis	2
Target Audience	2
Fee Model	2
Differences to Competitors	2
Marketing & Sales Plan	3
Design & Implementation	3
Design	3
Implementation	4
Testing & Performance Evaluation	9
Testing	9
Test Case Table	9
Unit Test Case Results	14
Black Box Testing	17
Equivalence Partitioning	17
Boundary Value Analysis	19
White Box Testing for Login	20
White Box Testing for Tax Calculator	21
Performance Evaluation	22
Overall Individual Contributions	22
Summary & Future Plans	24
References	25

Purpose of the Document

The purpose of this document is to highlight our fully implemented Customer Relationship Management (CRM) software system for finance professionals. We will discuss the background and insight that led to this project's completion. Furthermore, specifics on the technology & market analysis, overall differences to competitors, our fee model, marketing, system design, and test cases will also be provided. Finally, future implementations after the course and overall individual contributions will also be detailed.

Project Background

Our goal was to implement a CRM software system after an in-depth conversation with a Subject Matter Expert (SME). Upon doing research, we realized that a lack of CRMs specific to just financial professionals existed on the market. Only one-size-fits-all and generic solutions were available. Thus, we wanted to fill a void in the current market with our implementation, and create a tool that will assist self-employed individuals who do not have access to corporate tools.

We have successfully implemented such a tool, which allows self employed financial professionals, such as mortgage brokers, financial advisors, and insurance agents, to manage their clients, client appointments, generate reports, utilize calculators, and create expense forms. Our application was designed with the help of a finance professional's core job responsibilities in mind. Currently, professionals rely on multiple applications (either limited to their brokerage, or an independent one-size-fits-all CRM). Our software fixes this.

Our Subject Matter Expert (SME) indicated that to manage their clientele, the professional must have as much information on the client as possible. For anybody to be considered an advisor, they must familiarize themselves with the golden rule which is called Know Your Client (KYC). The software must have the capacity to have as much information on their clients. It should have a secure database and a user-friendly interface. All of which is implemented.

The application we have developed, took best practices from the tools the SME had used, and solved the issue of advisors having to familiarize themselves with multiple tools. The CRM we have successfully implemented isn't a one-size-fits-all solution like others available in the market; it focuses on the finance professionals core responsibilities while remaining effective, secure and user friendly.

Technology and Market Analysis

Target Audience

The target audience for our application is mortgage brokers, financial advisors, and insurance agents, as mentioned. Basically, any entity that would want to manage their own clientele. Upon research, we realized that there are over 200,000 licensed finance professionals in Canada. Therefore, this product can reach out to a large population, especially if they seek to manage their own clients. In regards to our product, we also see a potential for international expansion.

Both technology and self-employed financial professionals are projected to grow as we continue to evolve into a more online and technologically advanced world, thus, CRM software for financial professionals will be more sought after for individuals who want to manage their own clientele.

Fee Model

For our fee model, we chose to make our product a monthly subscription. For a single user, the cost would be \$30 per month, with a \$10 per month discount given for every additional user. So if there's two users, it will be \$50 per month instead of \$60. This is fair, as the average CRM system tends to cost around \$100 per month, with many going into the hundreds of dollars per month range.

Differences to Competitors

In our Design & Implementation section, our CRM system, as well as all of its features and functionality, will be showcased. There, it will be evident how our system differs from competitors. To summarize, the following are all of our differences from competitor CRMs:

- Simplistic and user friendly GUI
- Addition of the only most necessary and sought after features by financial professionals
- Competitor CRMs tend to be complex
- Expandability; easy to build on and implement your own features as it is entirely implemented with one language
- Local database storage as opposed to cloud storage for user convenience
- Ability to easily manage clients, their finances, schedule appointments for them, and generate client specific PDF reports
- Easy and convenient calculators
- Expense forms
- Cheaper; most CRMs tend to be costly to use, as many cost hundreds of dollars/month

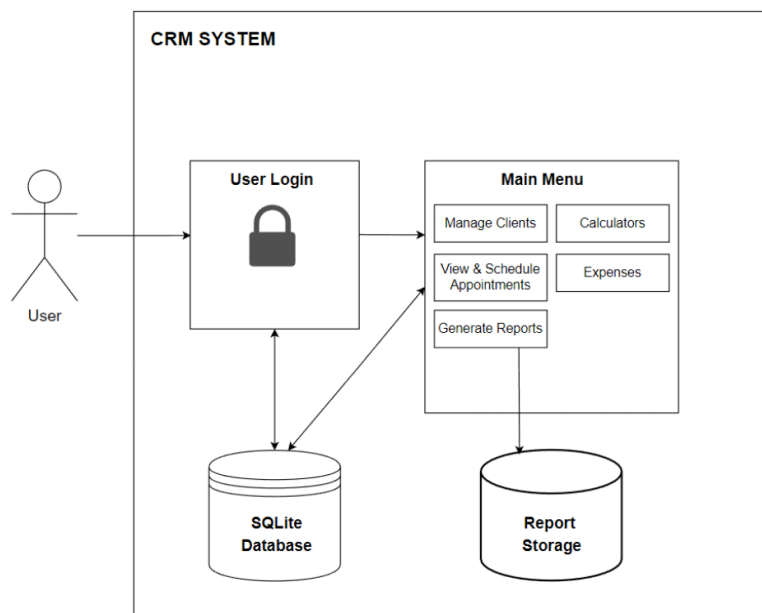
Marketing & Sales Plan

For our Marketing & Sales Plan, we would like to approach existing contacts to advertise our CRM system, such as brokerage managers, branch managers, and so forth. In addition, we would also like to run advertisements through various different channels. For instance, posting online advertisements, having our own website, and offering a free trial version so that users can test out the software before purchasing. We have also set out milestones for user acquisitions, as we are targeting 100 users by the end of year one.

In regards to funding, we are choosing to do bootstrapping, with our current team members filling various different organizational roles. This will allow us to have increased profitability in the early stages of our marketing. We can onboard investors at further stages if needed.

Design & Implementation

Design



Brief overview of the implementation:

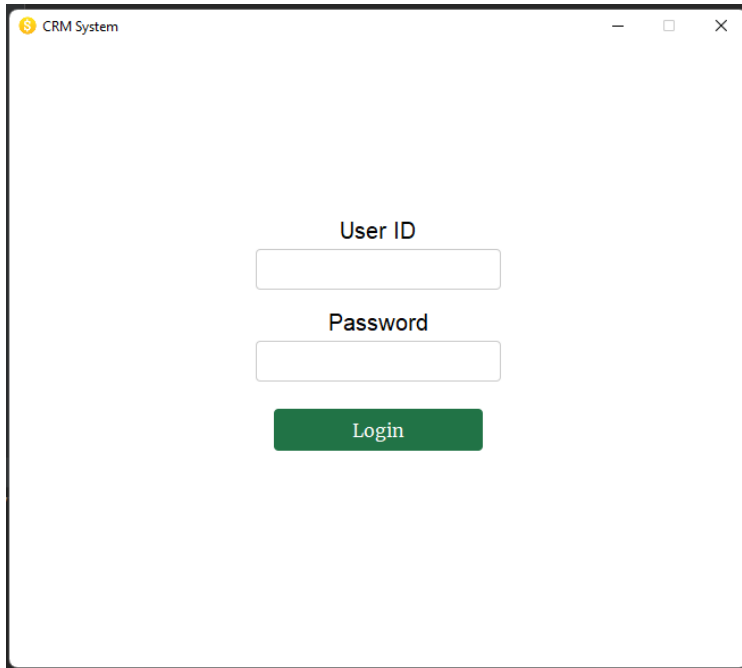
- IDE (PyCharm) for implementation
- All functionality coded with Python
- Database implemented with SQLite
- Login system incorporated with hash and salt password security to protect against threats
- Tkinter used for the GUI

Overall environment:

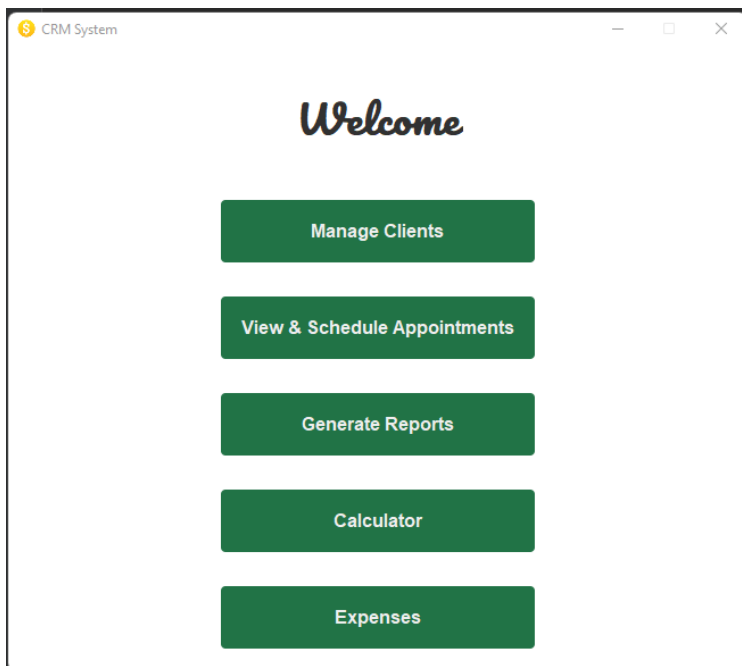
- Windows 10 & 11
- Python 3.10.4
- PyCharm 2022.1.2
- SQLite

Implementation

The application's focus is on being the hub that draws in the advisor's daily attention. We initialized the application with a log-in screen that focuses on security which uses hashing to increase the encryption complexity. Currently, the login & password we have set up is: abc (User ID), and 12345 (Password). This can be modified in the SQLite database.

A screenshot of a web application window titled "CRM System". The window contains a login form with two input fields: "User ID" and "Password". Below the "Password" field is a green "Login" button. The form is centered on a white background.

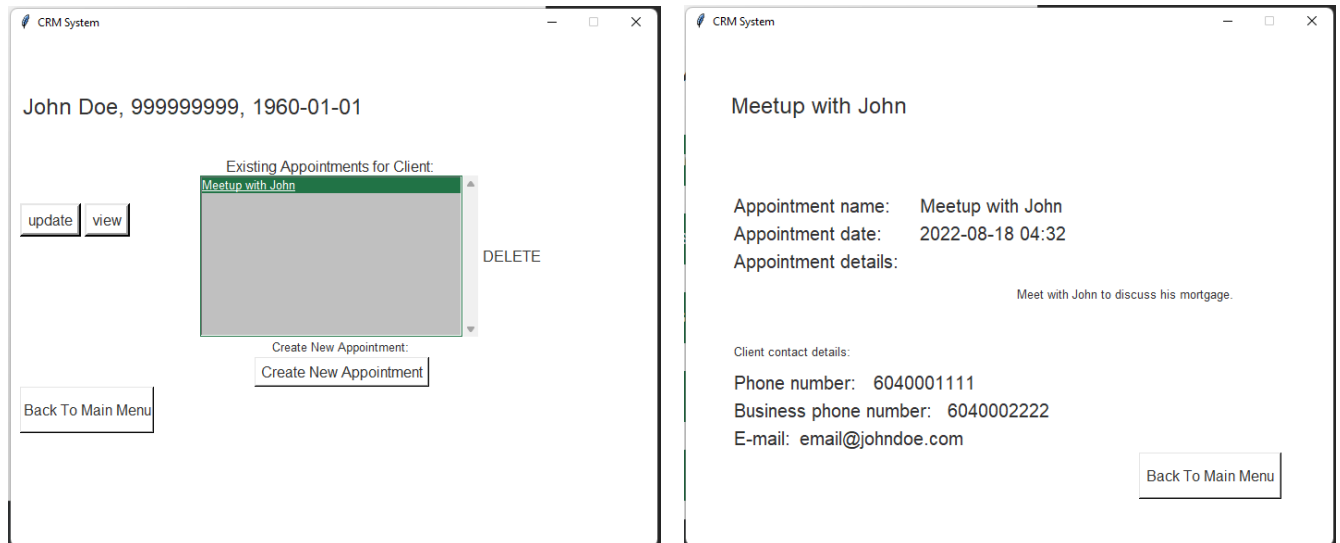
Once logged in with our secure log-in system, the user will be brought to the main page which will have all other subsystems that the users can access. This includes managing clients, scheduling appointments, generating reports, accessing different calculators, and adding expenses.

A screenshot of a web application window titled "CRM System". The window displays a "Welcome" message in a stylized font. Below the message are five green buttons arranged vertically: "Manage Clients", "View & Schedule Appointments", "Generate Reports", "Calculator", and "Expenses". The buttons are centered on a white background.

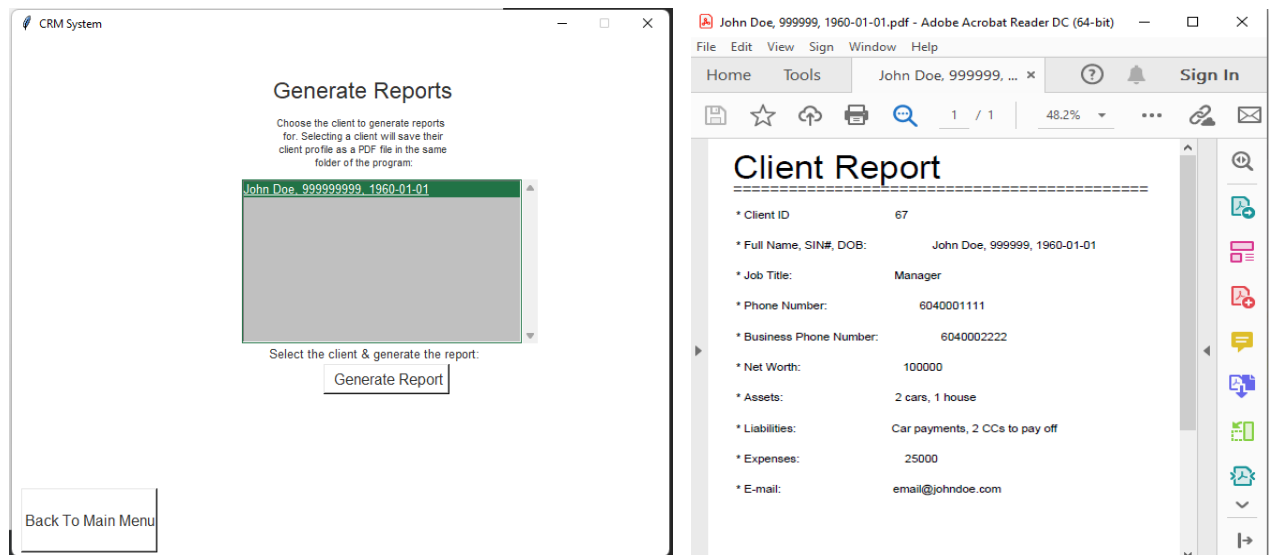
The managing clients subsystem allows users to view, update and create new clients. The application collects all vital information on the client for the advisor.

The view schedule and appointments subsystem does exactly as the name states. While booking the appointment, the advisor can enter key information so he/she can prepare adequately for the meeting. The application also auto-populates the client's information that exists on the database.

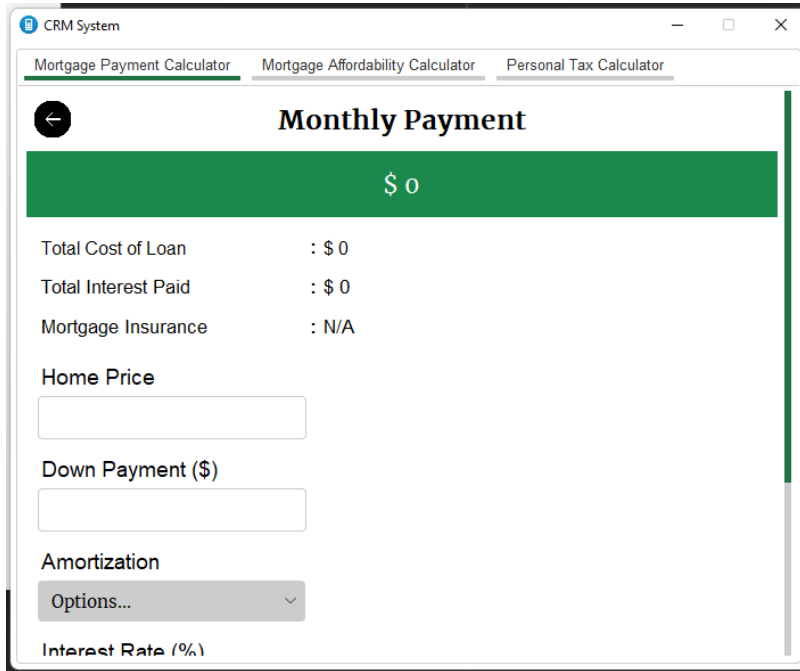
Once an appointment is created, the user can browse existing appointments for the selected client, update the existing appointment details, or view the appointment details.



The generate reports feature creates a PDF file for the chosen client and saves it locally, into the folder of where the application is.



There are three different calculators that the user can choose from, when clicking on the calculator subsystem: Mortgage Payment Calculator, Mortgage Affordability Calculator, and a Personal Tax Calculator.

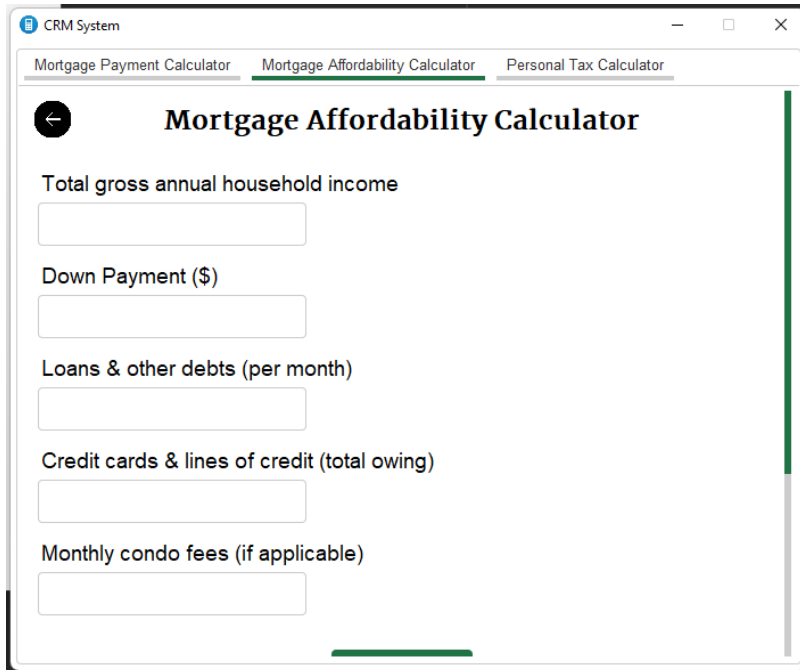


The screenshot shows a web application window titled "CRM System" with three tabs: "Mortgage Payment Calculator", "Mortgage Affordability Calculator", and "Personal Tax Calculator". The "Mortgage Payment Calculator" tab is active. The interface features a green header bar with a back arrow and the title "Monthly Payment". Below the header, a green box displays "\$ 0". The main content area includes a table with the following data:

Total Cost of Loan	: \$ 0
Total Interest Paid	: \$ 0
Mortgage Insurance	: N/A

Below the table, there are input fields for "Home Price" and "Down Payment (\$)". At the bottom, there is a section for "Amortization" with a dropdown menu labeled "Options..." and a label "Interest Rate (%)" below it.

The Mortgage Payment Calculator allows the user to put in a home price, the down payment, amortization period (1 year, up to 30 years), the interest rate, and the payment frequency (monthly, semi-monthly, or weekly). This will calculate the monthly payment for the home.



The screenshot shows the same "CRM System" window with the "Mortgage Affordability Calculator" tab active. The interface features a green header bar with a back arrow and the title "Mortgage Affordability Calculator". The main content area includes several input fields for the following information:

- Total gross annual household income
- Down Payment (\$)
- Loans & other debts (per month)
- Credit cards & lines of credit (total owing)
- Monthly condo fees (if applicable)

The Mortgage Affordability Calculator estimates mortgage affordability (maximum purchase price, and monthly mortgage payment), based on total gross annual household income, the down payment, and loans & debts and other fees.

The screenshot shows a web application window titled "CRM System" with three tabs: "Mortgage Payment Calculator", "Mortgage Affordability Calculator", and "Personal Tax Calculator". The "Personal Tax Calculator" tab is active. It features a back arrow icon, a title "Personal Tax Calculator", and a "Taxable Income" input field containing "50000". A green "Calculate" button is positioned below the input field. The results are displayed in a list:

Tax Payable	: \$ 7,481.09
After-Tax Income	: \$ 42,518.91
Average Tax Rate	: 14.96 %
Marginal Tax Rate	: 22.7 %
Marginal Rate on Capital Gains	: 11.35 %
Marginal Rate on Eligible Dividends	: 0.0 %
Marginal Rate on Ineligible Dividends	: 13.47 %

The Personal Tax Calculator calculates taxable income. For instance, a \$50,000 income has a tax payable amount of \$7,481.09, with an after-tax income of \$42,518.91 as shown, with the average tax rate being 14.96%.

The expenses subsystem allows the user to add expenses into the system. The user can choose the expense type (marketing, membership, fuel, gift, reimbursement, or other), as well as the expense amount, and the date. It will also add up the total expenses.

The screenshot shows a web application window titled "CRM System" with a single tab "Expenses". It features a title "Expenses" and three input fields labeled "Type", "Amount", and "Date". The "Type" field is a dropdown menu with "Other" selected. Below the input fields is a list of expense entries, each showing the type, amount, and date:

- other / 15.0 / 05052022
- Other / 123.0 / 1231
- Other / 123.0 / 1231
- Other / 123.0 / 1231
- Other / 2311.0 / 1231
- Other / 2311.0 / 1231
- Other / 2311.0 / 1231
- Other / 2311.0 / 1231
- Other / 12.0 / 1231
- Other / 12.0 / 1231

At the bottom left, the "Total" is displayed as 4326158.6. At the bottom right, there is an "Add Expense" button.

Testing & Performance Evaluation

Testing

Test Case Table

Test ID	Test Case Input	Test Condition	Result	Comments
1	Login			
1.1	User ID	Should be a valid userID	Pass	Displays an error saying "Incorrect username or password" if the user entered incorrect credentials.
1.2	Password	Should be a valid password	Pass	Displays an error saying "Incorrect username or password" if the user entered incorrect credentials.
2	Manage Clients			
2.1	Manage Client - Create a client			
2.1.1	Create a client form	All mandatory fields are filled	Pass	Displays an error "Fill all the fields" if all the required fields are not filled.
2.1.2	SIN number	9-digit number	Pass	Displays an error message if SIN is not a 9-digit number
2.1.3	DOB	Date format: YYYY-MM-DD	Pass	Displays an error message if the DOB entered is not in specified format
2.1.4	Phone Number	10-digit number	Pass	Displays an error message if the phone number is not a 10-digit number and ask the user to re-enter it.
2.1.5	Business Phone Number	10-digit number	Pass	Displays an error message if the phone number is not a 10-digit number and ask the user to re-enter it.
2.1.6	Email	Contains @ symbol	Pass	Displays an error message if email does not contain @ symbol and ask the user to re-enter it.
2.2	Manage Client – Update			
2.2.1	SIN number	9-digit number	Pass	Displays an error message if SIN is not a 9-digit number.
2.2.2	DOB	Date format: YYYY-MM-DD	Pass	Displays an error message if the DOB entered is not in specified format.

2.2.3	Phone Number	10-digit number	Pass	Displays an error message if the phone number is not a 10-digit number and ask the user to re-enter it.
2.2.4	Business Phone Number	10-digit number	Pass	Displays an error message if the phone number is not a 10-digit number and ask the user to re-enter it.
2.2.5	Email	Contains @ symbol	Pass	Displays an error message if email does not contain @ symbol and ask the user to re-enter it.
2.2.6	Update Client Form	Verify the client details are updated successfully	Pass	Displays "Success – updated" confirmation message and updates the client details.
2.3	Manage Client - View			
2.3.1	View Client details	Verify that the client details entered while creating a client are showing up on the view screen	Pass	
2.4	Manage Client – Delete			
2.4.1	Delete client	Verify that the client is deleted from the list of all clients	Pass	Displays "Success – deleted" confirmation message and deletes the client profile.
3	View & Schedule Appointments			
3.1	Create Appointments			
3.1.1	Create appointment form	Validate that all the mandatory fields are filled	Pass	Displays an error "Fill all the fields" if all the required fields are not filled.
3.1.2	Appointment Date	Validate the date is entered in YYYY-MM-DD H:M format	Pass	Displays an error message if the date is not in the specified format
3.1.3	Create appointment form	Validate the appointment is booked successfully	Pass	The appointment shows under "Existing Appointments for Client" section.

3.2	View Appointments			
3.2.1	View	Validate that the appointment details entered while booking an appointment are showing up on the view screen	Pass	
3.3	Update Appointments			
3.3.1	Appointment Date	Validate the date is entered in YYYY-MM-DD H:M format	Pass	Displays an error message if the date is not in the specified format
3.3.3	Update appointment form	Validate the appointment is updated successfully	Pass	Displays “Appointment was updated successfully” confirmation message and updates the appointment details.
3.4	Delete Appointment			
3.4.1	Delete	Verify that the appointment is successfully deleted from the list of all appointments.	Pass	Displays “Success – deleted” confirmation message and deletes the booked appointment.
4	Generate Reports			
	Generate Report	Verify that the report is generated successfully.	Pass	Generates a pdf report with the client details in the project folder.
5	Calculators			
5.1	Mortgage Payment Calculator			
5.1.1	Monthly payment	Calculates mortgage monthly payment using the information provided by the user	Pass	Verified the results by doing manual calculations and by running unit test.
5.1.2	Semi-monthly payment	Calculates mortgage semi-monthly payment using the information provided by the user	Pass	Verified the results by doing manual calculations and by running unit test.
5.1.1	Weekly payment	Calculates mortgage weekly payment using the information provided by the user	Pass	Verified the results by doing manual calculations and by running unit test.

5.1.2	Semi-monthly payment	Calculates mortgage semi-monthly payment using the information provided by the user	Pass	Verified the results by doing manual calculations and by running unit test.
5.1.1	Weekly payment	Calculates mortgage weekly payment using the information provided by the user	Pass	Verified the results by doing manual calculations and by running unit test.
5.2	Mortgage Affordability Calculator			
5.2.1	Total Debts	Calculates total debts by adding all the credit card, loans, condo fees etc.	Pass	Successfully adds up all the debts. Verified the results by doing manual calculations and by running unit test.
5.2.2	Monthly gross income	Calculates monthly income by dividing annual income by 12	Pass	Successfully calculates monthly income by using annual income. Verified the results by doing manual calculations and by running unit test.
5.2.3	Mortgage monthly payment	Calculated using the formula below: $PMT = 44\% \text{ of monthly income} - \text{total debts} - 500$ (heating, property tax etc.)	Pass	Successfully calculates mortgage monthly payment. Verified the results by doing manual calculations and by running unit test.
5.2.4	Mortgage Affordability	Calculates how much a person can afford. Calculated using the formula below:	Pass	Successfully calculates mortgage affordability amount. Verified the results by doing manual calculations and by running unit test.
5.3	Tax Calculator			
5.3.1	Provincial Tax	Provincial tax calculated depending upon which tax bracket the user income falls in	Pass	Successfully calculates provincial tax. Verified the results by doing manual calculations and by running unit test.

5.3.2	Federal Tax	Federal tax calculated depending upon which tax bracket the user income falls in	Pass	Successfully calculates federal tax. Verified the results by doing manual calculations and by running unit test.
5.3.3	Tax payable	Calculated using the formula below: Tax payable = Provincial tax + Federal tax	Pass	Successfully calculates tax payable. Verified the results by doing manual calculations and by running unit test.
5.3.4	After tax income	Calculated using the formula below: After tax income = Taxable income – Tax payable	Pass	Successfully calculates after-tax income. Verified the results by doing manual calculations and by running unit test.
5.3.5	Average tax rate	Calculated using the formula below: After tax rate = (Tax payable/ taxable income) x 100	Pass	Successfully calculates average tax rate. Verified the results by doing manual calculations and by running unit test.
5.3.6	Marginal tax rate	Calculated depending upon which tax bracket the user income falls in	Pass	Successfully calculates marginal tax rate. Verified the results by doing manual calculations and by running unit test.
5.3.7	Marginal rate on capital gains	Calculated depending upon which tax bracket the user income falls in	Pass	Successfully calculates marginal rate on capital gains. Verified the results by doing manual calculations and by running unit test.
5.3.8	Marginal rate on eligible dividends	Calculated depending upon which tax bracket the user income falls in	Pass	Successfully calculates marginal rate on eligible dividends. Verified the results by doing manual calculations and by running unit test.
5.3.9	Marginal rate on ineligible dividends	Calculated depending upon which tax bracket the user income falls in	Pass	Successfully calculates marginal rate on ineligible dividends. Verified the results by doing manual calculations and by running unit test.

6	Expenses			
	Expense	Allows to user to add expenses	Pass	Verified that the expense is saved successfully in the database, and it shows up in the list of expenses.

Unit Test Case Results

Test Create Client

The screenshot shows the PyCharm IDE with a test runner window open. The test runner displays the following information:

- Test Results:** 169 ms
- Tests passed:** 10 of 10 tests – 169 ms
- Command:** C:\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_create_client.MyTestCase
- Output:**

```

Testing started at 4:18 PM ...
Launching unittests with arguments python -m unittest test_create_client.MyTestCase in C:\Users\525802\PycharmProjects\finalProject

Process finished with exit code 0

Ran 10 tests in 0.175s

OK

```

The code editor on the left shows the `MyTestCase` class with the following test methods:

```

class MyTestCase(unittest.TestCase):

    def test_name_datatype(self):
        client = Createclient()
        full_name = client.entry_fullname.get()
        name_split = full_name.split(", ")
        name = name_split[0]
        self.assertFalse(name.isnumeric())

    def test_SIN_length(self):
        client = Createclient()
        full_name = client.entry_fullname.get()
        name_split = full_name.split(", ")
        sin = name_split[1]
        self.assertEqual(9, len(sin))

    def test_phone_length(self):
        client = Createclient()
        phone_number = client.entry_phonenumber.get()
        self.assertEqual(10, len(phone_number))

    def test_phone_datatype(self):
        client = Createclient()
        phone_number = client.entry_phonenumber.get()
        self.assertTrue(phone_number.isnumeric())

    def test_business_phone_length(self):
        client = Createclient()
        business_phone_number = client.entry_buisnessphonenumber.get()
        self.assertEqual(10, len(business_phone_number))

```

Test Update Client

The screenshot shows the PyCharm IDE with a test runner window open. The test runner displays the following information:

- Test Results:** 5 ms
- Tests passed:** 10 of 10 tests – 5 ms
- Command:** C:\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_update_client.MyTestCase
- Output:**

```

Testing started at 4:19 PM ...
Launching unittests with arguments python -m unittest test_update_client.MyTestCase in C:\Users\525802\PycharmProjects\finalProject

Ran 10 tests in 0.009s

OK

Process finished with exit code 0

```

The code editor on the left shows the `MyTestCase` class with the following test methods:

```

class MyTestCase(unittest.TestCase):

    def test_name_datatype(self):
        client = updateclient("John Doe, 999999999, 1960-01-01")
        full_name = self.client.entry_fullname.get()
        name_split = full_name.split(", ")
        name = name_split[0]
        self.assertFalse(name.isnumeric())

    def test_SIN_length(self):
        full_name = self.client.entry_fullname.get()
        name_split = full_name.split(", ")
        sin = name_split[1]
        self.assertEqual(9, len(sin))

    def test_phone_datatype(self):
        phone_number = self.client.entry_phonenumber.get()
        self.assertTrue(phone_number.isnumeric())

    def test_phone_length(self):
        phone_number = self.client.entry_phonenumber.get()
        self.assertEqual(10, len(phone_number))

    def test_business_phone_datatype(self):
        business_phone_number = self.client.entry_buisnessphonenumber.get()
        self.assertTrue(business_phone_number.isnumeric())

    def test_business_phone_length(self):
        business_phone_number = self.client.entry_buisnessphonenumber.get()
        self.assertEqual(10, len(business_phone_number))

```


Test New Appointment

```

import unittest
from new_appointment import CreateNewAppointment

class MyTestCase(unittest.TestCase):
    appt = CreateNewAppointment("John Doe, 999999999, 1960-01-01")

    def test_appt_name_is_null(self):
        # Test fails if the appointment name is null
        self.appt.entry_appointment_name.insert(0, "Test Name")
        appt_name = self.appt.entry_appointment_name.get()
        self.assertIsNotNone(appt_name)

    def test_appt_details_is_null(self):
        # Test fails if the appointment details field is null
        self.appt.entry_appointment_info.insert(0, "Test Details")
        appt_details = self.appt.entry_appointment_info.get()
        self.assertIsNotNone(appt_details)

if __name__ == '__main__':
    unittest.main()

```

Test Results

Tests passed: 2 of 2 tests - 1 ms

C:\Users\525802\AppData\Local\Programs\Python\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_new_appointment.MyTestCase

Testing started at 4:20 PM ...

Launching unittests with arguments python -m unittest test_new_appointment.MyTestCase in C:\Users\525802\PycharmProjects\finalProject

Ran 2 tests in 0.002s

OK

Process finished with exit code 0

Test Update Appointment

```

import unittest
from update_appointment import UpdateAppointment

class MyTestCase(unittest.TestCase):
    appt = UpdateAppointment("John Doe, 999999999, 1960-01-01", "67", "Appt 1")

    def test_appt_name_is_null(self):
        # Test fails if the appointment name is null
        appt_name = self.appt.entry_appointment_name.get()
        self.assertIsNotNone(appt_name)

    def test_appt_details_is_null(self):
        # Test fails if the appointment details field is null
        self.appt.entry_appointment_info.insert(0, "Test Details")
        appt_details = self.appt.entry_appointment_info.get()
        self.assertIsNotNone(appt_details)

if __name__ == '__main__':
    unittest.main()

```

Test Results

Tests passed: 2 of 2 tests - 1 ms

C:\Users\525802\AppData\Local\Programs\Python\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_update_appointment.MyTestCase

Testing started at 4:21 PM ...

Launching unittests with arguments python -m unittest test_update_appointment.MyTestCase in C:\Users\525802\PycharmProjects\finalProject

Ran 2 tests in 0.002s

OK

Process finished with exit code 0

Test Mortgage Payment Calculator

```

5 class MyTestCase(unittest.TestCase):
6     mortgage_calculator = calculator.calc()
7     mortgage_calculator.home_price.set(350000)
8     mortgage_calculator.down_payment.set(70000)
9     mortgage_calculator.amortization.set(25)
10    mortgage_calculator.interest_rate.set(4.69)
11
12    def test_mortgage_value(self):
13        home_price = int(self.mortgage_calculator.home_price.get())
14        down_payment = int(self.mortgage_calculator.down_payment.get())
15        self.assertEqual(280000, home_price - down_payment)
16
17    def test_monthly_payment(self):
18        self.mortgage_calculator.payment_frequency.set('Monthly')
19        monthly_payment = self.mortgage_calculator.PaymentCalculate()
20        self.assertEqual(1587, monthly_payment)
21
22    def test_semi_monthly_payment(self):
23        self.mortgage_calculator.payment_frequency.set('Semi-Monthly')
24        semi_monthly_payment = self.mortgage_calculator.PaymentCalculate()
25        self.assertEqual(792, semi_monthly_payment)
26
27    def test_weekly_payment(self):
28        self.mortgage_calculator.payment_frequency.set('Weekly')
29        weekly_payment = self.mortgage_calculator.PaymentCalculate()
30        self.assertEqual(365, weekly_payment)
31
32
33    if __name__ == '__main__':
34        unittest.main()

```

Run: Python tests for test_mortgage_payment_calculator.MyTestCase

Test Results

Tests passed: 4 of 4 tests - 3 ms

C:\Users\525802\AppData\Local\Programs\Python\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_mortgage_payment_calculator.MyTestCase

Testing started at 4:22 PM ...

Launching unittests with arguments python -m unittest test_mortgage_payment_calculator.MyTestCase in C:\Users\525802\PycharmProjects\finalProject

Ran 4 tests in 0.004s

OK

Process finished with exit code 0

Test Mortgage Affordability Calculator

```

5 class MyTestCase(unittest.TestCase):
6     mortgage_calculator = calculator.calc()
7     mortgage_calculator.annual_household_income.set(60000)
8     mortgage_calculator.down_payment2.set(50000)
9     mortgage_calculator.loans_other_debits.set(100)
10    mortgage_calculator.credit_cards.set(50)
11    mortgage_calculator.monthly_condo_fees.set(0)
12
13    def test_total_debts(self):
14        total_debts = float(self.mortgage_calculator.loans_other_debits.get()) + float(
15            self.mortgage_calculator.credit_cards.get()) + float(self.mortgage_calculator.monthly_condo_fees.get())
16        self.assertEqual(150, total_debts)
17        return total_debts
18
19    def test_monthly_gross_income(self):
20        monthly_income = float(self.mortgage_calculator.annual_household_income.get()) / 12
21        self.assertEqual(5000, monthly_income)
22        return monthly_income
23
24    def test_monthly_payment(self):
25        monthly_payment = self.mortgage_calculator.AffordabilityCalculate()[0]
26        self.assertEqual(1550, monthly_payment)
27
28    def test_mortgage_affordability(self):
29        mortgage_affordability = self.mortgage_calculator.AffordabilityCalculate()[1]
30        self.assertEqual(281214, mortgage_affordability)
31        return int(mortgage_affordability)
32
33    def test_home_affordability(self):
34        home_affordability = self.mortgage_calculator.AffordabilityCalculate()[2]

```

Run: Python tests for test_mortgage_affordability_calculator.MyTestCase

Test Results

Tests passed: 6 of 6 tests - 4 ms

C:\Users\525802\AppData\Local\Programs\Python\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_mortgage_affordability_calculator.MyTestCase

Testing started at 4:22 PM ...

Launching unittests with arguments python -m unittest test_mortgage_affordability_calculator.MyTestCase in C:\Users\525802\PycharmProjects\finalProject

Ran 6 tests in 0.007s

OK

Process finished with exit code 0

Test Tax Calculator

```

5 class MyTestCase(unittest.TestCase):
6     tax_calculator = calculator.calc()
7     tax_calculator.taxable_income.set(75000)
8     taxable_income = round(int(tax_calculator.taxable_income.get()), 2)
9
10    def test_tax_payable(self):
11        tax_payable = self.tax_calculator.TaxCalculator()
12        self.assertEqual(14520.05, tax_payable)
13        return round(tax_payable, 2)
14
15    def test_federal_tax(self):
16        federal_tax = self.tax_calculator._calculate_federal_tax(self.taxable_income)
17        self.assertEqual(10454.11, federal_tax)
18        return federal_tax
19
20    def test_provincial_tax(self):
21        provincial_tax = self.tax_calculator._calculate_provincial_tax(self.taxable_income)
22        self.assertEqual(4065.94, provincial_tax)
23        return provincial_tax
24
25    def test_tax_payable_as_sum(self):
26        tax_payable = round(self.test_federal_tax() + self.test_provincial_tax(), 2)
27        self.assertEqual(14520.05, tax_payable)
28
29    def test_after_tax_income(self):
30        after_tax_income = self.taxable_income - self.test_tax_payable()
31        self.assertEqual(60479.95, after_tax_income)
32
33    def test_average_tax_rate(self):
34        average_tax_rate = round((self.test_tax_payable() / self.taxable_income) * 100, 2)
35
36    MyTestCase.test_tax_payable()

```

Test Results

Tests passed: 10 of 10 tests - 14 ms

C:\Users\S25802\AppData\Local\Programs\Python\Python310\python.exe "C:\Program Files\JetBrains\PyCharm 2022.1\plugins\python\helpers\pycharm_jb_unittest_runner.py" --target test_tax_calculator.MyTestCase

Testing started at 4:24 PM ...

Launching unittests with arguments python -m unittest test_tax_calculator.MyTestCase in C:\Users\S25802\PycharmProjects\finalProject

Ran 10 tests in 0.018s

OK

Process finished with exit code 0

Black Box Testing

Equivalence Partitioning

- Validating the length of PhoneNumber for creating clients

PhoneNumber must be a positive 10-digit number.

The input value for the equivalence class partitioning is the length of PhoneNumber.

$\text{len}(\text{PhoneNum}) < 0$	$\text{len}(\text{PhoneNum}) < 10$	$\text{len}(\text{PhoneNum}) = 10$	$\text{len}(\text{PhoneNum}) > 10$
-----------------------------------	------------------------------------	------------------------------------	------------------------------------

TEST TABLE

Test ID	Input Value (Length of UserID)	Result	Partitions
1	-10	Error	$\text{len}(\text{PhoneNum}) < 0$
2	5	Error	$\text{len}(\text{PhoneNum}) < 10$
3	10	Valid	$\text{len}(\text{PhoneNum}) = 10$
4	12	Error	$\text{len}(\text{PhoneNum}) > 10$
5	"Tom"	Error	String input
6	11.25	Error	Decimal

- Calculating provincial (BC) tax in tax calculator

The amount of tax a person pays depends upon their total gross income and the amount varies depending upon which tax bracket their income falls in.

British Columbia tax bracket	British Columbia tax rate
Up to \$42,184	5.06%
\$42,185 to \$84,369	7.70%
\$84,370 to \$96,866	10.50%
\$96,867 to \$117,623	12.29%
\$117,624 to \$159,483	14.70%
\$159,484 to \$222,420	16.80%
Over \$222,420	20.50%

Test ID	Input Value (Income)	Result (BC Tax %)	Partitions
1	-10	Error	income < 0
2	10000	5.06%	0 <= income <= 42184
3	45000	7.70%	42184 < income <= 84396
4	94000	10.50%	84369 < income <= 96866
5	105000	12.29%	96866 < income <= 117623
6	125000	14.70%	117623 < income <= 159483
7	180500	16.80%	159583 < income <= 222420
8	235200	20.50%	222420 < income
9	"income"	Error	String input

- Calculating federal tax in tax calculator

Canadian federal tax bracket	Canadian federal tax rate
\$49,020 or less	15.00%
\$49,020 - \$98,040	20.50%
\$98,040 - \$151,978	26.00%
\$151,978 - \$216,511	29.00%
More than \$216,511	33.00%

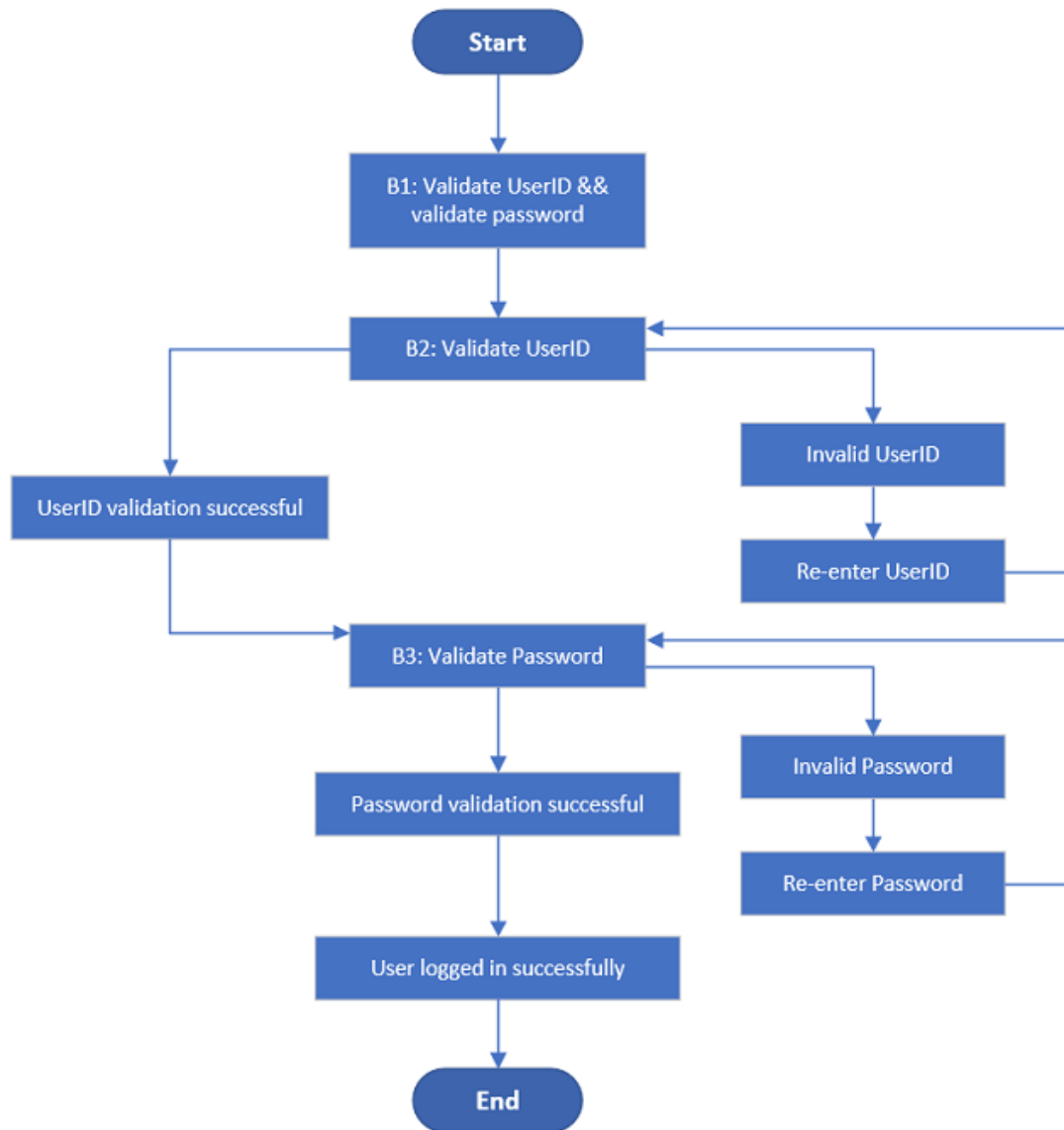
Test ID	Input Value (Income)	Result (BC Tax %)	Partitions
1	-10	Error	income < 0
2	10000	15.00%	0 <= income <= 49020
3	55000	20.50%	49020 < income <= 98040
4	105000	26.00%	98040 < income <= 151978
5	175000	29.00%	151978 < income <= 216511
8	235200	33.00%	216511 < income
9	"income"	Error	String input

Boundary Value Analysis

For Boundary Value Analysis, we made a table that includes valid testing input and invalid. This will save us major testing time. Minimum length of password is 6 and the maximum is 20. Anything under 6 and over 20 would be an invalid input.

Boundary Value Analysis		
Invalid	Valid	Invalid
(min-1)	(min, min+, nom, max-, max)	(max+1)
5	6, 7, 13, 19, 20	21

White Box Testing for Login



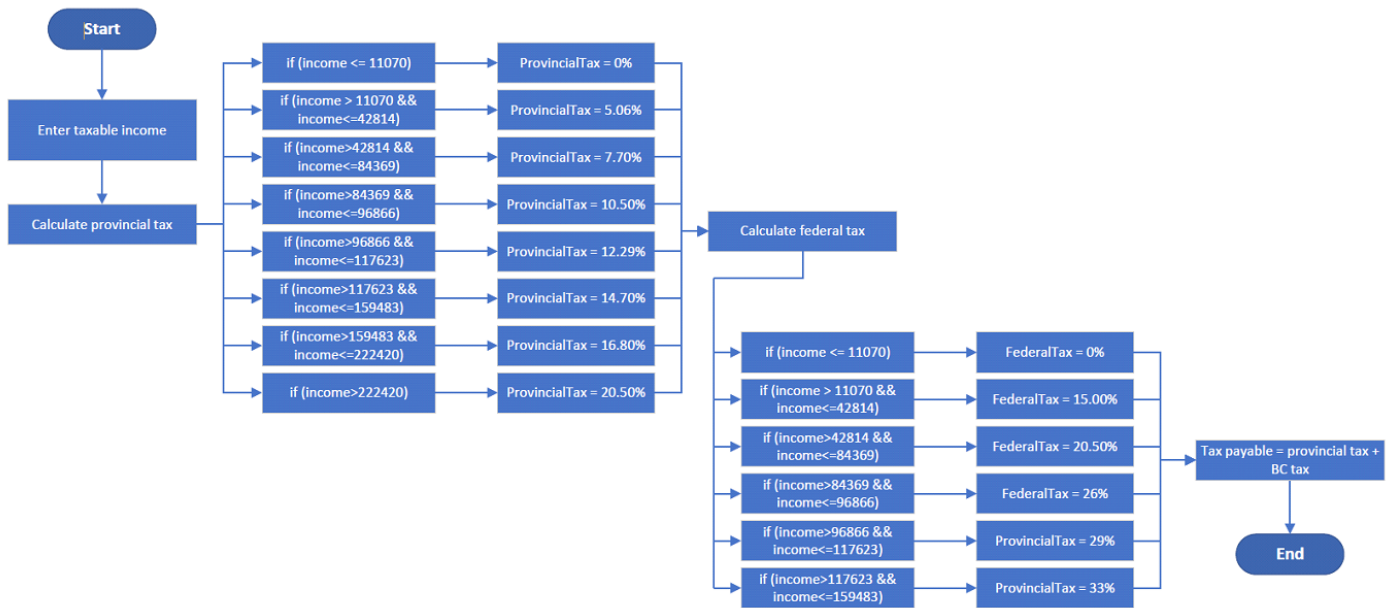
- Condition Coverage

	Condition 1	Condition 2	Decision
Test	validateUserId==1	validatePassword==1	if((validateUserId==1)&&(validatePassword==1))
T1 = (0,1)	F	T	F
T2 = (1,0)	T	F	F

- Decision Coverage

	Condition 1	Condition 2	Decision
Test	validateUserId ==1	validatePassword==1	if((validateUserId==1)&&(validatePassword==1))
T1 = (0,1)	F	T	F
T2 = (1,0)	T	F	F

White Box Testing for Tax Calculator



Performance Evaluation

We have tested our fully implemented CRM system adequately to ensure that the response time and loading speed are in check. Moreover, we have induced stress on the system to ensure that it can handle information overload. Additionally, we ensured that the system can handle as many clients as necessary, and that the database can handle queries as needed. We have also tested the login system to ensure optimal functionality, speed, and security.

Overall Individual Contributions

I (**Harman Dhaliwal**), was responsible for a lot of the subsystems that were implemented in our fully functional CRM system, acting as the principal programmer for this project. Moreover, I was also in charge of writing a large portion of the final report (common part), and putting together the final PPT slides, and coordinating the presentations and editing the videos.

My team member, Jas Mangat, acted as the Project Manager and SME, assisting with code implementation, some documentation as well (midterm & summary report), and being in charge of implementing a few of the subsystems, such as the login system, and expenses. Mansimar Kaur also acted as a programmer, as well as quality assurance, implementing some vital tools such as the calculators, and theming the system. Moreover, she was responsible for the final testing and final unit test cases in our documents. Lastly, Neil Maedel acted as the programming support. Helping us implement our subsystems together seamlessly, providing encryption for our database, implementing SQL error checking, and assisting with various different systems, like the login and expenses. Furthermore, he was responsible for the midterm PPT slides.

As mentioned in my midterm presentation video portion, and presentation slides, the subsystems I was responsible for implementing, were the client management and client profile portions of the system, the appointment scheduling feature, the database for the client and appointment information in SQLite, the PDF reports generation feature, and finally, the GUI using Tkinter. I also provided assistance/ideas for the other subsystems that were developed by my group members - the login system, calculators, and expense forms.

In the following figure, I will showcase the files that I implemented and coded, as part of our full implementation. Then, I will offer a timeline for when all of these functionalities were implemented.

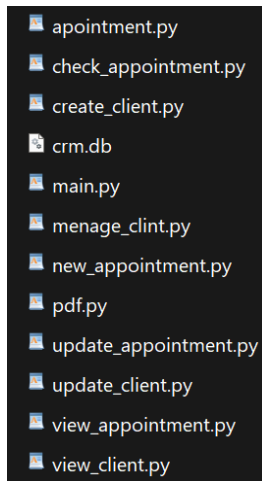


Figure on the left:

The Python code files I implemented, including the “crm.db” SQLite database which stores the necessary client and appointment information. The program is run through launching the “main.py” file in a Python IDE application (PyCharm in our case). Tkinter, FPDF, and various other libraries need to be installed to run the code without any errors. The final overall code for our system will be attached alongside the final report and presentation in the submission.

Timeline (my individual contributions, not including team members):

- May 10th - 13th: Researched GUI implementation using Python and completed the GUI implementation using Tkinter
- May 14th - 19th: Worked on the client profile UI and SQLite database for storing our client information
- May 19th - 21st: Worked on, and completed the overall client management subsystem (all of the GUI windows, viewing, creating, modifying, and deleting clients)
- May 22nd - 24th: Worked on, and modified the SQLite database to expand its functionality for the upcoming appointment scheduling feature
- May 24th - 30th: Worked on, and completed the overall appointment scheduling subsystem (all of the GUI windows, viewing, creating, modifying, and deleting appointments)
- May 31st - June 1st: Modified the SQLite database to allow for PDF client creation to function appropriately
- June 2nd - June 6th: Implemented the front-end for PDF client creation
- June 18th - June 27th: Assisted with the midterm PPT slides and merged and submitted the midterm presentation video
- July 8th - July 10th: Worked on individual contributions for midterm report
- July 11th - August 1st: Assisted group members with other subsystems (login, calculators, expenses)
- August 1st - August 8th: Set up, and put together the final PPT slides, coordinated the final presentation, and merged and submitted the final presentation video
- August 8th - August 14th: Worked on the common part of the final report, as well as documented my overall individual contributions

Summary & Future Plans

To summarize, we have implemented a CRM system that includes the following functionalities and features:

- Fully implemented GUI using Tkinter
- Fully implemented encrypted login system with hash and salt password security
- Fully implemented the client management & client profile section
- Fully implemented the database for the client and appointment information in SQLite
- Fully implemented the PDF reports generation feature of the client
- Fully implemented a mortgage payment calculator
- Fully implemented a mortgage affordability calculator
- Fully implemented a tax calculator
- Fully implemented expense forms
- Fully implemented SQL error checking

In regards to future implementations, we plan on expanding the functionality of our system further, by implementing tools such as mass email, contact management, and notes. In addition, we would also like to work further on the user interface. For example, to allow the user to customize the modules. Finally, we would like to work on the software licensing and copyright to ensure that we can sell our product legally.

References

- Best, R. (n.d.). *The 5 best CRM software for Financial Advisors in 2022*. Investopedia. Retrieved August 10, 2022, from <https://www.investopedia.com/best-crm-software-for-financial-advisors-5084287>
- Chipman, S. (2022, May 8). *What does CRM actually cost?* CRM Switch. Retrieved August 11, 2022, from <https://crmswitch.com/buying-crm/crm-cost-list/>
- Goldstein, B. (2017, May 16). *How much does CRM cost?: Prices of 37 leading CRMs*. How Much Does CRM Cost? | Prices of 37 Leading CRMs. Retrieved August 11, 2022, from <https://www.nutshell.com/blog/how-much-does-crm-cost>
- How to find your audience using CRM*. CRM Software - Streamlined CRM Solutions Dubai. (2020, March 19). Retrieved August 13, 2022, from <https://www.saphyte.com/blog/marketing/how-to-find-your-audience-using-crm>
- Onyett, C. (2021, January 7). *How to market software products in 5 effective ways [guide]*. Rocketto. Retrieved August 13, 2022, from <https://www.hellorocketto.com/articles/how-to-market-software>
- Randall, S. (2017, November 24). *Canada's financial services sector is growing fast*. Wealth Professional. Retrieved August 10, 2022, from <https://www.wealthprofessional.ca/news/industry-news/canadas-financial-services-sector-is-growing-fast/234548>
- Wowa. (2022, May 30). *BC Income Tax Calculator*. WOWA.ca. Retrieved August 14, 2022, from <https://wowa.ca/bc-tax-calculator>