UNIVERSITY OF NAIROBI


**FACULTY OF SCIENCE AND TECHNOLOGY**

**DEPARTMENT OF COMPUTING AND INFORMATICS**


# MARANGI PAINTING MANAGEMENT SYSTEM


**MUTUA RYAN SILU**

**P15/2100/2021**

**SUPERVISOR**

**PROFESSOR PETER WAGACHA**


**A second-year project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science in Computer Science of the University of Nairobi**

# ABSTRACT

The Marangi Painting Management System is a comprehensive solution designed to connect customers with skilled painters and streamline the management of painting jobs. This system aims to address the challenges faced by customers in finding reliable painters and by painters in sourcing jobs efficiently. The system facilitates the registration of customers and painters, allowing clients to post painting projects and painters to create job proposals. It has 2 modules – Client Module and Painter Module. It provides features such as job allocation, contract creation, invoice generation, and revenue calculation. The system ensures secure data storage and retrieval, enabling painters to maintain accurate records of their customers and payments. Additionally, it fosters improved communication between painters and clients, allowing for periodic updates on job progress. By offering an efficient and user-friendly platform, the Marangi Painting Management System aims to revolutionize the painting industry and provide a seamless experience for both customers and painters.

# Table of Contents

ABSTRACT ..... 1
DECLARATION ..... 5
DEDICATION ..... 6
ACKNOWLEDGEMENTS ..... 6
INTRODUCTION ..... 1
Background Information: ..... 1
Problem Statement: ..... 2
Research and System Development Objectives: ..... 3
Scope: ..... 4
Assumptions: ..... 4
Justification: ..... 5
LITERATURE REVIEW ..... 6
Introduction ..... 6
Companies that have solved similar problems: ..... 6
Crown Paints ..... 6
Jiji ..... 6
Upwork ..... 6
Critique ..... 7
Conclusion ..... 7
SYSTEMS ANALYSIS and DESIGN ..... 8
Requirements Elicitation: ..... 8
Functional Requirements: ..... 8
Non-functional Requirements: ..... 9
Constraints: ..... 10
Deliverables: ..... 12
Target Environment: ..... 12
Task Schedule: ..... 13
Gantt Chart ..... 14
System Analysis ..... 15
Methodology: ..... 15
Dataflow Diagram: ..... 16
Data Flow Descriptions: ..... 17
a) Painter or Client Registration: ..... 17

Level 1 Diagram: 17
Data Structures: 19
b) Job and Proposal Creation 26
Level 1 Diagram: 26
Data Structures: 28
c) Create and Sign Contract 36
Level 1 Diagram 36
Data Structures 37
d) Job Completion, Earnings, and Ratings 41
Level 1 Diagram 41
Data Structures 42
Database Design 44
Site Map: 46
User Flows: 47
SYSTEM IMPLEMENTATION 49
Resources Used: 49
Technologies Used: 50
ReactJS: 50
Python Flask and SQLAlchemy: 50
Visual Studio Code (VSCode): 50
Insomnia: 50
.env file: 50
Karla font: 50
Google Cloud's Geolocation API: 50
Testing 51
Frontend optimization: 51
Localhost configuration: 51
User Testing: 51
CONCLUSION 52
References 53
APPENDIX A: USER MANUAL 54
Home Pages 54
Client 56
APPENDIX B: Sample Code 71
Python Flask 71

React JS ..... 72
APPENDIX C: Sample Contract ..... 74
Annex: ..... 75
Interview Guide: ..... 75
Painter Interview Guide: ..... 75
Customer Interview Guide: ..... 76

# DECLARATION

I hereby declare that this project is my own work, and has to the best of my knowledge, not been submitted to any other institution of higher learning.

**Student:** Mutua Ryan Silu **Reg No:** P15/2100/2021

**Signature**: _____

**Date**: .....

This project has been submitted as a partial fulfilment of the requirements for the Bachelor of Science in Computer Science of the University of Nairobi with my approval and that of my Project Supervisor.

**Supervisor:** Professor Peter Waiganjo Wagacha

**Signature**: _____

**Date**: .....

## DEDICATION

This dedication is a sincere acknowledgment and appreciation for the professors, lecturers, faculty members, and colleagues who have played a crucial role in bringing the project to fruition. Their unwavering dedication provided valuable feedback, support, and inspiration, motivating me to successfully complete the project. Their support and insights have been pivotal in shaping the project's success. This dedication stands as a tribute to their belief, challenges, and unwavering support, which have transformed this project into a reality and sparked further accomplishments.

## ACKNOWLEDGEMENTS

I would like to express deep gratitude to my incredible family, whose steadfast support and understanding were essential throughout my journey. Their encouragement and love have fueled my determination to reach my goals. I am also immensely thankful to my dedicated project supervisor, whose guidance and expertise have been invaluable throughout this undertaking. Their insights and mentorship have played a significant role in shaping the success of this project.

Additionally, I would like to extend my appreciation to my exceptional colleagues who have been by my side every step of the way. Their encouragement and camaraderie have made this journey not only productive but also enjoyable.

# INTRODUCTION

## Background Information:

The Marangi Painting Management System is aimed at helping customers get access to quality painters within their location easily and assisting the painting companies manage their jobs efficiently. The system will allow companies or individual painters to manage the painting jobs to be done, each job having several tasks to be performed. The system will allow customers to register, post their painting jobs or project, allocate the jobs to a painter within their proximity who they view to be the best candidate for the job. They can see the ratings of the painters and their previous works, getting a taste of what their buildings may look like when they are painted. It will store information gathered in a database for easier retrieval, addition, and deletion, create contracts, generate invoices for payments and registration, and calculate revenue made by individual painters from all the payments made monthly and annually.

Construction owners require their buildings wholly painted or a portion of the building to be repainted after some time. The buildings vary from residential apartments, individual homes, hotels, malls, and others, each with varying number of units. The owners have been accustomed to searching for painters through word of mouth whereby they outsource from friends who have had their buildings painted by the painters. They have little information on other work done by the painters and are normally not sure of the skill level of the painters until their buildings are painted.

Painters on the other hand do not easily find potential painting jobs. Apart from being recommended to by a satisfied customer to another customer, they normally get a tip from a friend of an ongoing construction or come across a construction site to scout and bid for the painting job once the construction is finished. They also work together with some contractors and are among the subcontractors hired to perform a certain task within the construction cycle, theirs being painting. Once they start the jobs, they manually manage and document the amount they will spend and what they will earn, both gross and net.

## Problem Statement:

There are customers who hire subcontractor painters to perform specific painting tasks on building projects such as apartments, individual houses, workshops, and commercial buildings. Most of the painters normally take a lot of time outsourcing jobs and may require the contractors to get the jobs and hire them if they need some extra work when not occupied with their own tasks. They may have issues keeping a deadline due to lack of enough planning based on the timeframe given by the client. Many of their key system functionalities are manual, but some are blended with software such as excel for record keeping and mobile services for payments.

Customers on the other hand mainly know how to source the painters through word of mouth from their counterparts who have painted buildings, get phone numbers of the potential painters, and come to an agreement. Some do get them from social media platforms or websites but those are few. The building contractors for these customers may also have their own painter(s) in the crew they hire.

There are some cases where clients and painters do not sign agreements and have a verbal agreement on what is to be done by the painter and the resources. It may lead to breach of contract between either of the parties when unsatisfied, leaving the job undone, or a painter underpaid.

Problems with the current system are as follows:

- Customers do not majorly find skilled painters within their area, that have a concrete portfolio of some of the jobs they have done other than by word of mouth and referrals.
- Customers do not have most painters to source from fast and will have between 1 and 3 to choose from.
- Painters take time before sourcing jobs.
- Hard to keep track of the amount of time to be spent on a job and the deadlines.
- Painters lack an efficient computerized system to store and maintain data of their customers. For example, it is easy to confuse the amount paid by a customer of a one painting job from that of another if not keen enough since it must be documented manually first.
- Data can be easily lost since it is recorded in tangible files without a database to be stored.
- Minimal data security.
- It is difficult to update the customer the progress of their jobs periodically and allow the painter to update the progress.
- It is tedious for the painter and clients to know which of their jobs have been completed, and those ongoing. The client also may have a tougher time knowing which jobs have not yet started and do not have a contracted painter.

## Research and System Development Objectives:

The primary objectives of this project are to:

- Develop a user-friendly and efficient Painting Management System that can be used by painters and clients to manage painting jobs and connect them to each other.
- Provide a platform for easy:
  a) Registration of Clients
  b) Registration of Painters
  c) Registration of Jobs
  d) Creation of Job Proposals by painters
  e) Creation of Contracts by clients
  f) Rating of Painters by clients
- Enable easy access to job details and payment history via contracts for both painters and customers.
- Ensure secure storage and easy retrieval of:
  a) Job Details
  b) Contract Details
  c) Proposal Details
  d) Painter and Client Details
- Provide an automated system for generating contracts and calculating revenue.
- Continuous testing and deployment: Continuous testing and deployment has been implemented to ensure that the system is stable and secure. This involves the use of automated testing tools and continuous integration and delivery tools to identify and fix issues in the development cycle.

## Scope:

The scope of this project includes the development of a user-friendly Painting Management System that facilitates the connection between clients and painters, allowing them to manage painting jobs efficiently. The system will enable clients to register and post their painting projects, while painters can register and create job proposals. It will include features such as job allocation, contract creation, invoice generation, and revenue calculation. The system will store and manage data related to job details, contracts, proposals, painters, and clients in a secure database. Additionally, the system will provide functionality for rating painters by clients and tracking the progress of ongoing jobs.

## Assumptions:

This project assumes that the user can access the internet by use of a mobile device, a desktop computer, or a laptop.

## Justification:

The Marangi Painting Management System aims to address several existing challenges in the current process of hiring and managing painters for building projects. The justification for this project includes:

i) Improved Painter Selection:
The system will enable clients to access a pool of skilled painters within their area and make informed decisions based on the painters' portfolios, ratings, and reviews.

ii) Enhanced Efficiency:
Both clients and painters will benefit from an efficient platform that streamlines the process of job allocation, contract creation, and progress tracking. This will save time and effort for all parties involved.

iii) Accurate Data Management:
By utilizing a computerized system, painters can maintain accurate records of their customers, jobs, and payments. This eliminates the risk of data loss and confusion between different jobs.

iv) Secure Data Storage:
The system will ensure secure storage of sensitive data, protecting the privacy and confidentiality of both painters and clients.

v) Improved Communication:
The system will provide a means for painters to update clients on the progress of their jobs, fostering better communication and transparency throughout the painting process.

vi) Streamlined Financial Management:
With automated contract generation and revenue calculation, painters can easily track their earnings from various jobs, improving financial management and record-keeping.

vii) Scalability and Growth:
The system can be scaled and expanded to accommodate a growing number of painters and clients, making it a sustainable solution for the long term.

Overall, the Marangi Painting Management System addresses the limitations of the current process, enhances efficiency, promotes transparency, and provides a reliable platform for clients and painters to connect and manage painting jobs effectively.

# LITERATURE REVIEW

## Introduction

The Marangi Painting Management System aims to revolutionize the process of connecting clients with skilled painters and efficiently managing painting jobs. To better understand the current landscape and identify effective solutions, this literature review explores similar companies that have successfully addressed similar problems.

## Companies that have solved similar problems:

### Crown Paints

Crown Paints offers a website that facilitates the connection between customers and painters within the same county based on their location (Crown, 2022). The platform provides a pool of painters ready to undertake painting projects for various parts of buildings. Customers enter their details and location, and upon agreeing to the terms and conditions, they are presented with a webpage displaying painters near them. The webpage provides relevant information about each painter, including ratings, profiles, team size, and project handling capacity. Customers can then directly contact the chosen painter to discuss the specific requirements of their painting project.

### Jiji

Jiji.com is an online platform that enables individuals to sell goods and advertise services, including painting services (Jiji, 2022). Painters can create an account on Jiji, showcasing their works and providing details about their services. This platform allows customers to browse through painter profiles, view their previous works, and contact them directly to discuss their painting needs.

### Upwork

Upwork is a popular platform that connects customers with freelancers across various industries, including painting (Upwork, 2022). Painters can create profiles on Upwork and bid on painting jobs posted by customers. Customers review the proposals received from different painters, conduct interviews if necessary, and ultimately select the best fit for their project.

## Critique

The reviewed companies have successfully tackled the challenge of connecting customers with painters. Crown Paints stands out as a niche-specific platform solely dedicated to the painting field. Upwork excels in customer-painter relationship management, offering customers detailed painter profiles, portfolios of previous work, and the ability to communicate within the platform before finalizing a hire. Customers can also post job requirements and receive bids from different painters, enabling them to choose the most suitable painter for their project.

However, there are still opportunities for improvement in the existing systems. The Marangi Painting Management System aims to address these shortcomings by providing a comprehensive and tailored solution that combines the strengths of these platforms. By offering enhanced features such as efficient job allocation, contract creation, progress tracking, and secure data management, the Marangi system aims to streamline the painting management process and provide a more seamless experience for both customers and painters.

## Conclusion

The reviewed companies have made significant strides in connecting customers with painters and addressing the challenges in the painting industry. However, there is a need for an integrated and customized system like the Marangi Painting Management System to further enhance the efficiency, transparency, and overall experience of managing painting jobs. By learning from the strengths and weaknesses of existing platforms, the Marangi system aims to provide a comprehensive solution that caters specifically to the needs of both customers and painters in the industry.

# SYSTEMS ANALYSIS and DESIGN

## Requirements Elicitation:

### Functional Requirements:

- The system registers clients and painters separately by providing their name, property, email address, and phone number.
- The system allows the clients to register jobs by providing the name, location, customer, painting area, starting date, and deadline.
- Painters can see recently posted jobs on their side with those within their jurisdiction appearing first.
- A job stops being displayed to other painters once:
  a) A contract has been created based off it.
  b) It has stayed without being bid for 30 days.
  c) Has reached the number of proposals possible as entered by the client.
  d) Its proposed end date by the client has passed.
- Clients can choose the painter they see fit for the job based on the information they get about the painter and select or deselect a proposal by any painter.
- Clients can see the portfolios of the painters who have bid for the job.
- A client enters the maximum number of proposals for a job to prevent the job from being bombarded with proposals.
- A client can confirm the selected proposal for a job by creating a contract with the painter. If they agree on the price, it is added to the template of the contract created and the painter-client agreement is generated.
- The client enters the price per unit having specified whether the job is labor-based or labor and material-based.
- A client can cancel the selected proposal for a job – If the customer and painter do not come to an agreement, the customer can cancel the proposal and look for another painter.
- A client can delete a job before a contract has been created for it.
- A painter can delete a proposal before it is selected by a client.
- A painter can create a portfolio for the clients to see their past works and decide whether they want them to paint their buildings.
- To complete a job both the painter and client must confirm that the job is completed and the
- The system provides a way to store and maintain data in a database for easier retrieval, addition, and deletion.
- The system must provide a way to sign the painter-client agreement in a transparent and legal way.

- Clients can distinguish between the jobs that have not got a painter yet, those ongoing, and those completed.
- Clients can rate the painters after they complete their jobs out of 5 stars.
- The system calculates revenue made by the painters from all the payments made monthly and annually.
- Painters can see their average ratings and see the number of jobs per rating number.
- The painter can create a plan to work with within the deadline as they see the number of days left per job.
- Painters and Clients can print the contract they signed digitally, and sign physically with a magistrate's signature for legal emphasis in case it is required.

## Non-functional Requirements:

- The system must be user-friendly and easy to use.
- The system must be reliable and available 24/7.
- The system must be scalable and able to handle many users and jobs.
- The system must have a high level of security to protect sensitive information.
- The system must perform well and be responsive.
- The system must be maintainable and updatable.

## Constraints:

The following constraints have been taken into consideration when designing and developing the system:

- The system has been developed using technologies that are accessible and easy to use for both the painter and the customer.
- The system will be scalable to accommodate additional features in the future. Such features include:
    - Mobile App: Creating a mobile app that allows painters to view their assigned jobs and update their progress can increase efficiency and productivity. Additionally, allowing customers to view the status of their jobs through the app can improve transparency and communication.
    - A new Payments feature in the app where a painter can see how much they owe the web application.
    - Clients choose whether to pay via the web application or directly to the painter.

- The system has been designed with data security in mind to prevent unauthorized access to customer information and payment details. It incorporates features such as use of access and refresh tokens for authorization.
- The system is developed with the assumption that some users may have limited experience using computerized systems.
- The system has been developed within a budget and can use readily available hardware and software.
- User roles and permissions: The system has different user roles with specific permissions. For example, painters have access to their assigned jobs only, while customers see different painters that bid for a job. This will ensure data security and minimize the risk of data breaches.
- Integration with third-party services: The system will eventually be designed to integrate with third-party services such as payment gateways, email services, and SMS services. This will improve the user experience and reduce the workload of the system. For example, the system can be connected to a specific bank account where the payments are made to, or use of M-Pesa. SMSs can be used to send confirmation of payments and completion of projects before the invoices are created.
- System monitoring and logging: The system has monitoring and logging capabilities to identify and resolve issues before they become critical. The backend receives alerts when critical events occur, and logs should be maintained for future analysis.
- Data backup and recovery: Since data loss is a concern with the current system, the new system should have a data backup and recovery plan. The plan should include regularly scheduled backups

and a procedure for restoring data in case of system failure. Data is stored in both a local database and the cloud. Backup can also be stored in nonvolatile storage devices such as hard disks and solid-state devices.

## Deliverables:

The following deliverables will be produced as part of the project:

a) Requirements Specification Document - This document will outline the functional and non-functional requirements of the system.
b) Data Flow Diagrams - These will be used to represent the interactions between the user and the system.
c) System Architecture Diagram - This will provide a high-level overview of the system's design and components.
d) Source Code - The source code for the frontend and backend components of the system.
e) Database Schema - This will describe the database structure and how the different entities relate to each other.
f) User Manual - This will provide a guide for users on how to use the system.
g) Testing Plan - This will provide a roadmap for the testing and validation of the system.
h) Maintenance Plan - This will outline how the system will be maintained and updated in the future.

## Target Environment:

System Environment:

The target environment for the painting management system is a web-based application that can be accessed from any device with an internet connection, such as desktop computers, laptops, tablets, and smartphones.

The system should be responsive and easy to navigate and use on any device by adapting to different screen sizes.

It should be suitable to use on most popular web browsers such as Google Chrome, Apple's Safari and Mozilla Firefox.

The system should be deployed on a secure server with appropriate security measures in place to ensure the confidentiality and integrity of user data.

User Target Environment:

It is aimed at painters and clients who want to get painting jobs and get painters respectively.

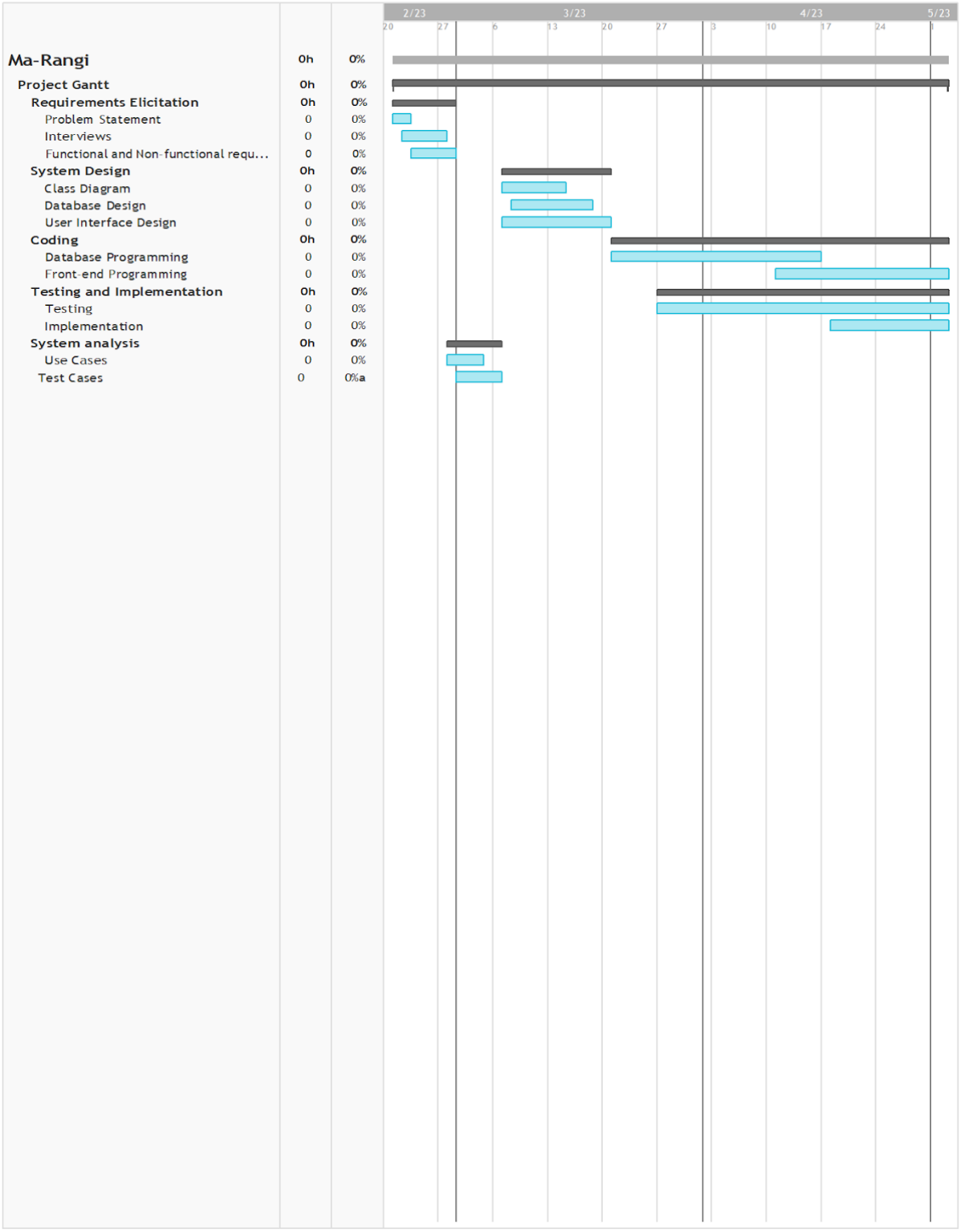
## Task Schedule:

The following was the tentative schedule followed for the development of Marangi Painting Management System:

- Requirements elicitation and information gathering – Week 1 – Week 2
- Systems Analysis – Week 3
- Systems Design – Week 4
- Coding – Week 5 - 9
- Testing and Implementation – Week 10

## Gantt Chart

teamgantt
Created with Free Edition



## System Analysis

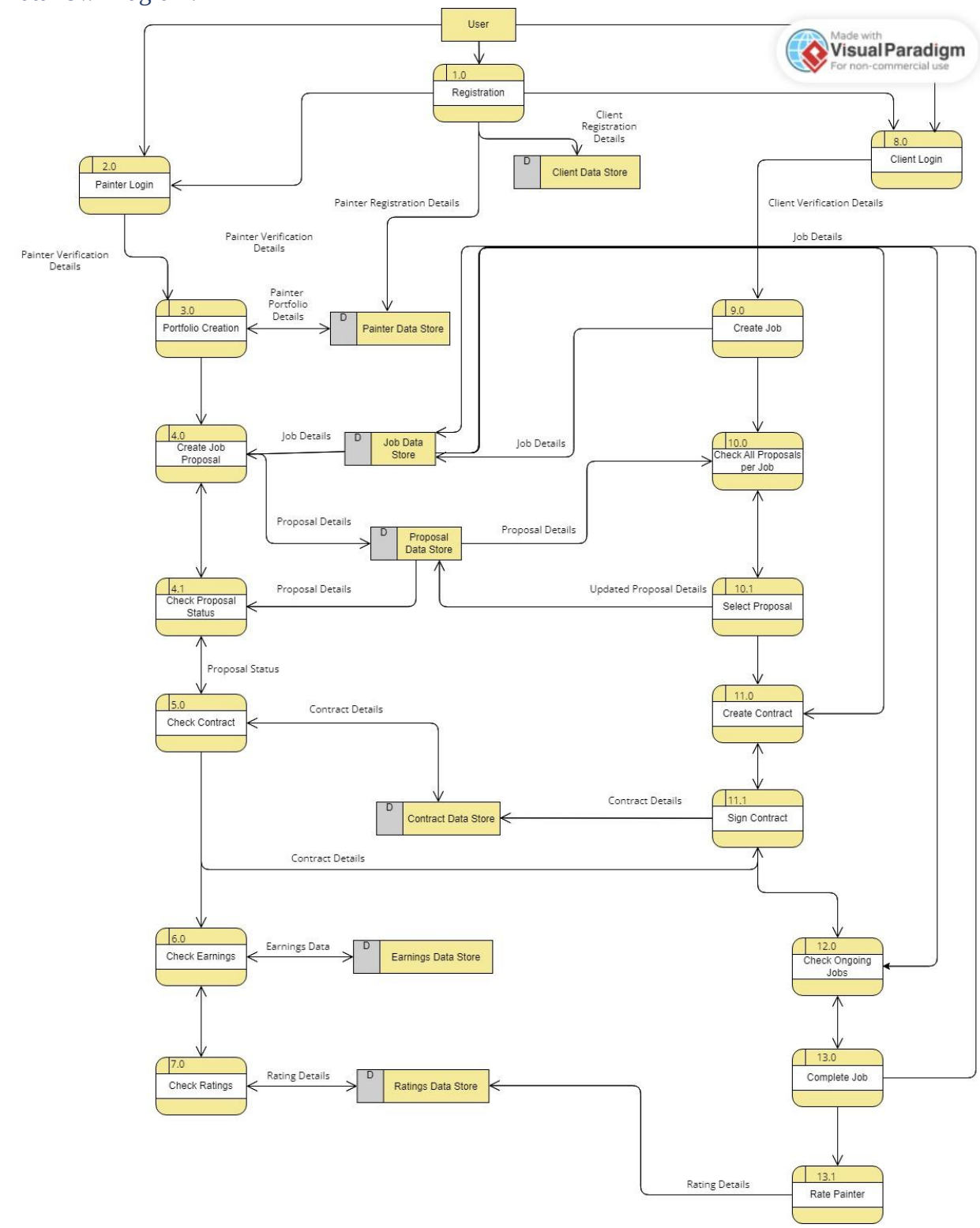
### Methodology:

The Project has been created using the Agile Methodology. The methodology allowed the system to interact with the users not only when it is fully developed but also when it is under development. The system is delivered gradually, with the requirements, plan and outcomes being checked regularly to respond to change faster.

The painting management system has been a good fit to this methodology due to the following reasons:

- The system interacted with the painters and clients on a regular basis to understand what they needed and what was not needed in the project allowing it to adapt to their requirements.
- It allowed actualizing the coding, analysis, and design aspects together such that after analyzing, and designing a certain part of the system for example a register painter use case, it can be coded with the outcome passed to some of the interviewed painters and others in the same space to evaluate.
- It allowed flexibility and adaptability in project development.
- It helped modify the documentation of the system with every change instead of having a whole documentation after completion of the project and making numerous changes after.
- Working software was the main metric for progress.
- Having face-to-face interactions with the painters and clients built a rapport and made them the first customers of the product.

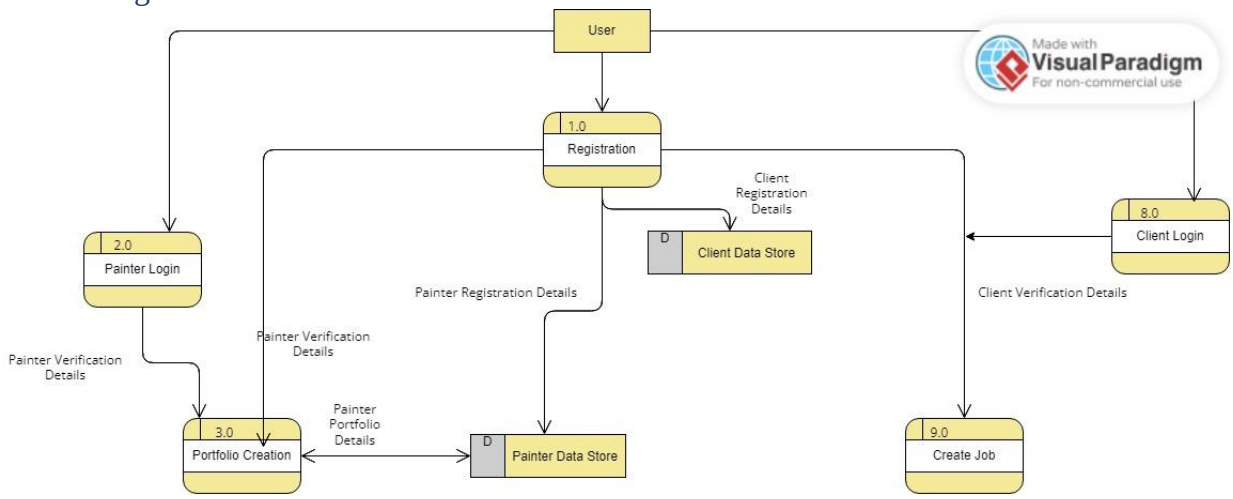
## Dataflow Diagram:



## Data Flow Descriptions:

a) Painter or Client Registration:

### Level 1 Diagram:




1. Painter Registration Details

| | |
|---|---|
| ID | 1.0 |
| Name | Painter Registration Details |
| Description | Has details of the painter |
| Source | User |
| Destination | Painter Data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Painter Details |

2. Client Registration Details

| | |
|---|---|
| ID | 1.0 |
| Name | Client Registration Details |
| Description | Has details of the client |
| Source | User |
| Destination | Client Data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Client Details |

3. Painter Portfolio Data:

| | |
|---|---|
| ID | 3.0 |
| Name | Painter Portfolio Details |
| Description | Has details of the painter portfolio |
| Source | Painter |
| Destination | Painter Data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Painter Portfolio |

4. Painter Verification Details:

| | |
|---|---|
| ID | 1.0, 2.0 |
| Name | Painter Verification Details |
| Description | Has details painter needs to access the system: - Email and Password |
| Source | Painter |
| Destination | Portfolio Creation |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Painter Verification Details |

5. Client Verification Details:

| | |
|---|---|
| ID | 1.0, 8.0 |
| Name | Client Verification Details |
| Description | Has details client needs to access the system: - Email and Password |
| Source | Client |
| Destination | Create Job |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Client Verification Details |

## Data Structures:

### 1. *Painter Details:*

Painter Details = ID + First Name + Last Name + Gender + Email + Password + Phone Number + Area + Painter Created At

*Element Description:*

ID:

| Name | ID |
|---|---|
| Description | Registration ID of the painter |
| Data Type | Integer |
| Validation | Not Null |
| Primary Key | True |

First Name:

| Name | First_name |
|---|---|
| Description | First name of the painter |
| Data Type | String |
| Validation | Not Null |
| Length | 20 |

Last Name:

| Name | last_name |
|---|---|
| Description | Last name of the painter |
| Data Type | String |
| Validation | Not Null |
| Length | 20 |

### Gender:

| Name | gender |
|---|---|
| Description | Gender of the painter |
| Data Type | Enum -> (Male, Female) |
| Validation | Not Null |
| Length | 6 |

### Email:

| Name | email |
|---|---|
| Description | Email of the painter |
| Data Type | Text |
| Validation | Not Null |
| Length | 80 |
| Unique | True |

### Password:

| Name | password |
|---|---|
| Description | Password for painter authentication |
| Data Type | Text |
| Validation | Not Null |
| Length | 30 |

### Area:

| | |
|---|---|
| Name | area |
| Description | Area that the painter resides in that helps match the locale of the painter with that of the job |
| Data Type | Enum -> Dagoretti North, Dagoretti South, Embakasi Central, Embakasi East, Embakasi North, Embakasi South, Embakasi West, Kamukunji, Kasarani, Kibra, Langata, Makadara, Mathare, Nairobi Central, Roysambu, Ruaraka, Starehe, Westlands |
| Validation | Not Null |
| Length | 20 |

### Phone Number:

| | |
|---|---|
| Name | Phone_number |
| Description | Painter's Phone Number |
| Data Type | String |
| Validation | Not Null |
| Length | 15 |

### Painter Created At:

| | |
|---|---|
| Name | Painter_created_at |
| Description | Get the time the new Painter was created |
| Data Type | DateTime |
| Validation | Not Null |
| Default | server_default=func.now() -> Actual Date and Time at that particular moment |

### *2. Client Details:*

Client Details = ID + First Name + Last Name + Gender + Email + Password + Phone Number + Client Created At

*Element Description:*

ID:

| | |
|---|---|
| Name | ID |
| Description | Registration ID of the Client |
| Data Type | Integer |
| Validation | Not Null |
| Primary Key | True |

First Name:

| | |
|---|---|
| Name | First_name |
| Description | First name of the Client |
| Data Type | String |
| Validation | Not Null |
| Length | 20 |

Last Name:

| | |
|---|---|
| Name | last_name |
| Description | Last name of the Client |
| Data Type | String |
| Validation | Not Null |
| Length | 20 |

Gender:

| | |
|---|---|
| Name | gender |
| Description | Gender of the Client |
| Data Type | Enum -> (Male, Female) |
| Validation | Not Null |
| Length | 6 |

### Email:

| | |
|---|---|
| Name | email |
| Description | Email of the Client |
| Data Type | Text |
| Validation | Not Null |
| Length | 80 |
| Unique | True |

### Password:

| | |
|---|---|
| Name | password |
| Description | Password for Client authentication |
| Data Type | Text |
| Validation | Not Null |
| Length | 30 |

### Phone Number:

| | |
|---|---|
| Name | Phone_number |
| Description | Client's Phone Number |
| Data Type | String |
| Validation | Not Null |
| Length | 15 |

### Client Created At:

| | |
|---|---|
| Name | Painter_created_at |
| Description | Get the time the new Client was created |
| Data Type | DateTime |
| Validation | Not Null |
| Default | server_default=func.now() -> Actual Date and Time at that particular moment |

### *3. Painter Portfolio Data*

Painter Portfolio = ID + Portfolio Short Code + Portfolio Created At + Description + PainterID

*Element Description*

ID:

| Name | ID |
|---|---|
| Description | Registration ID of the Portfolio |
| Data Type | Integer |
| Validation | Not Null |
| Primary Key | True |

Portfolio Short Code:

| Name | Portfolio_Short_Code |
|---|---|
| Description | A random 4-digit short code that uniquely identifies a portfolio by a painter used to abstract the actual position of the portfolio in the database |
| Data Type | String |
| Validation | Not Null |
| Length | 4 |
| Unique | True |

Portfolio Created At:

| Name | Portfolio_created_at |
|---|---|
| Description | Get the time the new Portfolio was created |
| Data Type | DateTime |
| Validation | Not Null |
| Default | server_default=func.now() -> Actual Date and Time at that particular moment |

### Description:

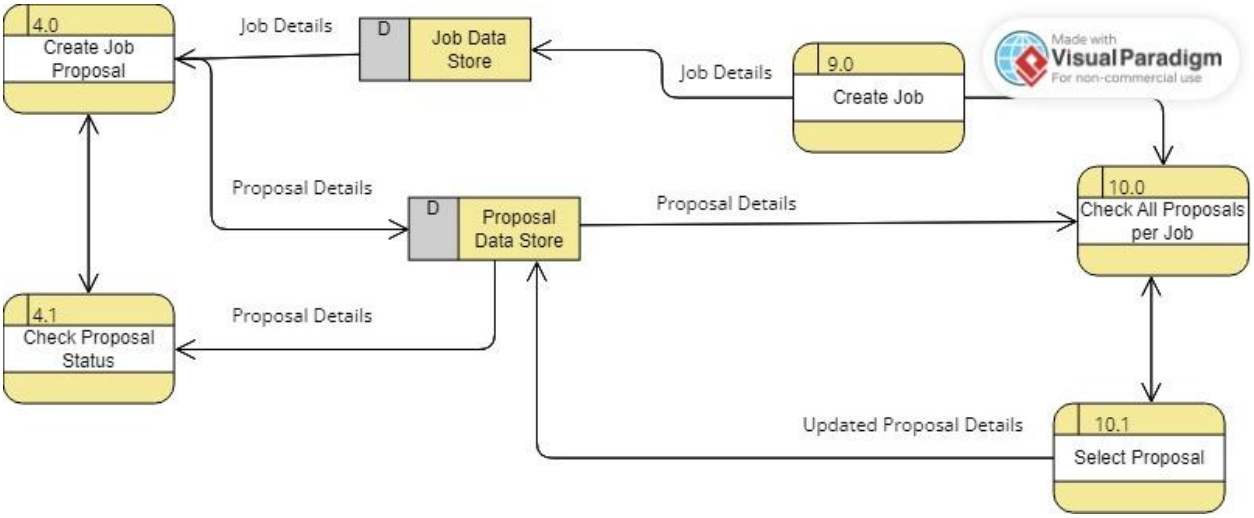
| Name | Description |
|---|---|
| Description | Painter's description of his / her portfolio |
| Data Type | String |
| Validation | Not Null |
| Length | 2500 |

### Painter ID:

| Name | Painter_ID |
|---|---|
| Description | Painter_ID for the painter owning the specific portfolio |
| Data Type | Integer |
| Validation | Not Null |
| Foreign Key | True |
| | |

## b) Job and Proposal Creation

### Level 1 Diagram:




1. Job Details:

| | |
|---|---|
| ID | 9.0 |
| Name | Job Details |
| Description | Describes the details of a particular job. |
| Source | Client |
| Destination | Job Data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Job Details |

| | |
|---|---|
| ID | 4.0 |
| Name | Job Details |
| Description | Describes the details of a particular job. |
| Source | Job Data Store |
| Destination | Painter -> Create Job Proposal |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Job Details |

2. Proposal Details

| | |
|---|---|
| ID | 4.0 |
| Name | Proposal Details |
| Description | Describes the details of a particular proposal to a particular job. |
| Source | Create Job Proposal |
| Destination | Proposal data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Proposal Details |

| | |
|---|---|
| ID | 4.1, 10.0 |
| Name | Proposal Details |
| Description | Describes the details of a particular proposal to a particular job. |
| Source | Proposal data Store |
| Destination | Check Proposal Status, Check All Proposals per Job |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Proposal Details |

3. Updated Proposal Details

| | |
|---|---|
| ID | 10.1 |
| Name | Updated Proposal Details |
| Description | When Client selects a proposal, its selected status changes hence the update |
| Source | Select Proposal |
| Destination | Proposal data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Proposal Details |

## Data Structures:

### 1. *Job Details:*

Job Details = ID + Job Short Code + Job Name + Job Description + Property Location + Property Type + Job Type + Contract Type + Total Floors + Total Rooms + Start Date + End Date + Job Confirmed + Job Completed + Max Proposals + Rated + Client ID

*Element Description:*

ID:

| Field | Value |
|---|---|
| Name | ID |
| Description | Registration ID of the Job |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

Job Short Code:

| Field | Value |
|---|---|
| Name | Job Short Code |
| Description | A random 4-digit short code that uniquely identifies a job (starts with 4 digits) |
| Data Type | String |
| Validation | Not Null |
| Length | 4 |

Job Name:

| Field | Value |
|---|---|
| Name | Job Name |
| Description | Name of the Job |
| Data Type | String |
| Validation | Not Null |
| Length | - |

### Job Description

| Field | Value |
|---|---|
| Name | Job Description |
| Description | Description of the Job |
| Data Type | String |
| Validation | Not Null |
| Length | - |

### Property Location

| Field | Value |
|---|---|
| Name | Property Location |
| Description | Location of the property |
| Data Type | Enum |
| Validation | Not Null |
| Length | - |

### Property Type

| Field | Value |
|---|---|
| Name | Property Type |
| Description | Type of the property |
| Data Type | Enum |
| Validation | Not Null |
| Length | - |

### Job Type

| Field | Value |
|---|---|
| Name | Job Type |
| Description | Type of the job |
| Data Type | Enum |
| Validation | Not Null |
| Length | - |

### Contract Type

| Field | Value |
|---|---|
| Name | Contract Type |
| Description | Type of the contract |
| Data Type | Enum |
| Validation | Not Null |
| Length | - |

### Total Floors

| Field | Value |
|---|---|
| Name | Total Floors |
| Description | Total number of floors in the property |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

### Total Rooms

| Field | Value |
|---|---|
| Name | Total Rooms |
| Description | Total number of rooms in the property |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

### Start Date

| Field | Value |
|---|---|
| Name | Start Date |
| Description | Start date of the job |
| Data Type | Date |
| Validation | Not Null |
| Length | - |

### End Date

| Field | Value |
|---|---|
| Name | End Date |
| Description | End date of the job |
| Data Type | Date |
| Validation | Not Null |
| Length | - |

### Job Confirmed

| Field | Value |
|---|---|
| Name | Job Confirmed |
| Description | Indicates if the job is confirmed |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |

### Job Completed

| Field | Value |
|---|---|
| Name | Job Completed |
| Description | Indicates if the job is completed |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |

### Client ID

| Field | Value |
|---|---|
| Name | Client ID |
| Description | ID of the client associated with the job |
| Data Type | Integer |
| Validation | - |
| Length | - |

### Maximum Proposals

| Field | Value |
|---|---|
| Name | Max Proposals |
| Description | Maximum number of proposals allowed for the job |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

### Rated

| Field | Value |
|---|---|
| Name | Rated |
| Description | Indicates if the job has been rated |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |

### Job Created At

| Field | Value |
|---|---|
| Name | Job Created At |
| Description | Date and time when the job was created |
| Data Type | DateTime |
| Validation | Not Null |
| Length | - |

### *2. Proposal Details*

Proposal Details = ID + Proposal Short Code + Proposal Date + Proposal Name + Proposal Description + Proposal Selection + Proposal Confirmation + Job ID + Painter ID

*Element Description*

ID

| Field | Value |
|---|---|
| Name | ID |
| Description | Registration ID of the Proposal |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

Proposal Short Code

| Field | Value |
|---|---|
| Name | Proposal Short Code |
| Description | A unique short code that identifies the proposal |
| Data Type | String |
| Validation | Not Null |
| Length | - |
| Unique | True |

Proposal Date

| Field | Value |
|---|---|
| Name | Proposal Date |
| Description | Date and time when the proposal was created |
| Data Type | DateTime |
| Validation | Not Null |
| Length | - |

### Proposal Name

| Field | Value |
|---|---|
| Name | Proposal Name |
| Description | Name of the proposal |
| Data Type | Text |
| Validation | Not Null |
| Length | - |

### Proposal Description

| Field | Value |
|---|---|
| Name | Proposal Description |
| Description | Description of the proposal |
| Data Type | Text |
| Validation | Not Null |
| Length | - |

### Proposal Selection

| Field | Value |
|---|---|
| Name | Proposal Selection |
| Description | Indicates if the proposal has been selected |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |
| Default | False |

### Proposal Confirmed

| Field | Value |
|---|---|
| Name | Proposal Confirmed |
| Description | Indicates if the proposal has been confirmed |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |
| Default | False |

### Job ID

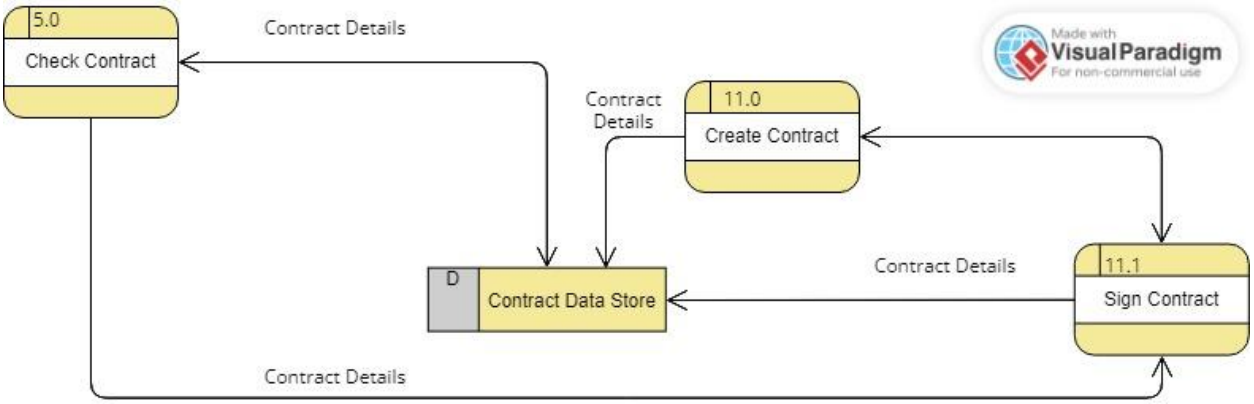
| Field | Value |
|---|---|
| Name | Job ID |
| Description | ID of the job associated with the proposal |
| Data Type | Integer |
| Validation | - |
| Length | - |

### Painter ID

| Field | Value |
|---|---|
| Name | Painter ID |
| Description | ID of the painter associated with the proposal |
| Data Type | Integer |
| Validation | - |
| Length | - |

## c) Create and Sign Contract

### Level 1 Diagram




1. Contract Details:

| | |
|---|---|
| ID | 5.0 |
| Name | Contract Details |
| Description | Describes the details of a particular contract to a particular job. |
| Source | Contract Data Store or Check Contract |
| Destination | Contract Data Store or Check Contract |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Contract Details |

| | |
|---|---|
| ID | 11.0, 11.1 |
| Name | Contract Details |
| Description | Describes the details of a particular contract to a particular job. |
| Source | Sign Contract |
| Destination | Contract Data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Contract Details |

## Data Structures

### *1. Contract Details*

Contract Details = ID + Contract Short Code + Materials + Exterior Lumpsum + Interior preparation + Interior Finishing + Total Payment Amount + Client Sign + Painter Sign + signed + signed At + Job ID

#### Element Description:

#### ID

| Field | Value |
|---|---|
| Name | ID |
| Description | Registration ID of the Contract |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

#### Contract Short Code

| Field | Value |
|---|---|
| Name | Contract Short Code |
| Description | A unique short code that identifies the contract |
| Data Type | String |
| Validation | Not Null |
| Length | - |
| Unique | True |

#### Materials

| Field | Value |
|---|---|
| Name | Materials |
| Description | List of materials for the contract |
| Data Type | String |
| Validation | - |
| Length | - |

### Exterior Lumpsum

| Field | Value |
|---|---|
| Name | Exterior Lumpsum |
| Description | Lump sum amount for exterior work |
| Data Type | Integer |
| Validation | - |
| Length | - |

### Interior Preparation

| Field | Value |
|---|---|
| Name | Interior Preparation |
| Description | Amount for interior preparation work |
| Data Type | Integer |
| Validation | - |
| Length | - |

### Interior Finishing

| Field | Value |
|---|---|
| Name | Interior Finishing |
| Description | Amount for interior finishing work |
| Data Type | Integer |
| Validation | - |
| Length | - |

### Total Payment Amount

| Field | Value |
|---|---|
| Name | Total Payment Amount |
| Description | Total payment amount for the contract |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

### Client Sign

| Field | Value |
|---|---|
| Name | Client Sign |
| Description | Indicates if the client has signed the contract |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |
| Default | False |

### Painter Sign

| Field | Value |
|---|---|
| Name | Painter Sign |
| Description | Indicates if the painter has signed the contract |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |
| Default | False |

### Signed

| Field | Value |
|---|---|
| Name | Signed |
| Description | Indicates if the contract has been signed |
| Data Type | Boolean |
| Validation | Not Null |
| Length | - |
| Default | False |

### Signed At

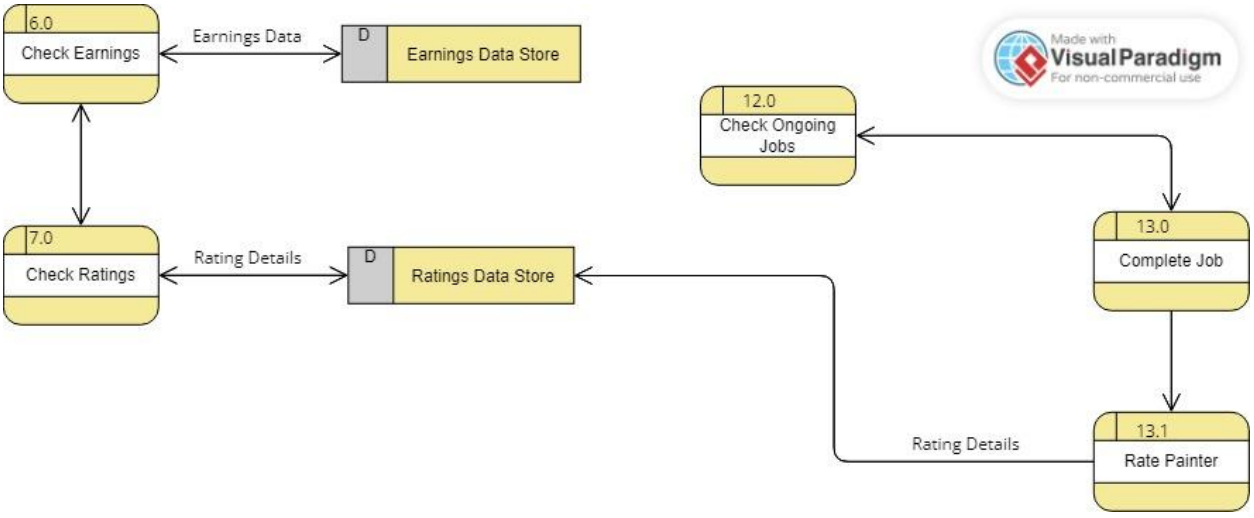
| Field | Value |
|---|---|
| Name | Signed At |
| Description | Date and time when the contract was signed |
| Data Type | DateTime |
| Validation | - |
| Length | - |

### Job ID

| Field | Value |
|---|---|
| Name | Job ID |
| Description | ID of the job associated with the contract |
| Data Type | Integer |
| Validation | - |
| Length | - |

## d) Job Completion, Earnings, and Ratings

### Level 1 Diagram




1. Rating Details:

| | |
|---|---|
| ID | 7.0, 13.1 |
| Name | Rating Details |
| Description | Has details for a random rating given by a client to the work done by a player |
| Source | Rate Painter, Check Ratings |
| Destination | Check Ratings, Ratings Data Store |
| Type of Data Flow | Screen |
| Data Structure travelling with Data Flow | Rating Details |

2. Earnings Data

It is sorted from contracts signed by the painter, with the total sum of all earnings being queried and shown either per month or year.

## Data Structures

### 1. *Rating Details*

Rating Details = ID + Rating Short Code + Rating No + Painter ID

#### *Element Description*

#### ID

| Field | Value |
|---|---|
| Name | ID |
| Description | Registration ID of the Rating |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

#### Rating Short Code

| Field | Value |
|---|---|
| Name | Rating Short Code |
| Description | A unique short code that identifies the rating |
| Data Type | String |
| Validation | Not Null |
| Length | - |
| Unique | True |

#### Rating No

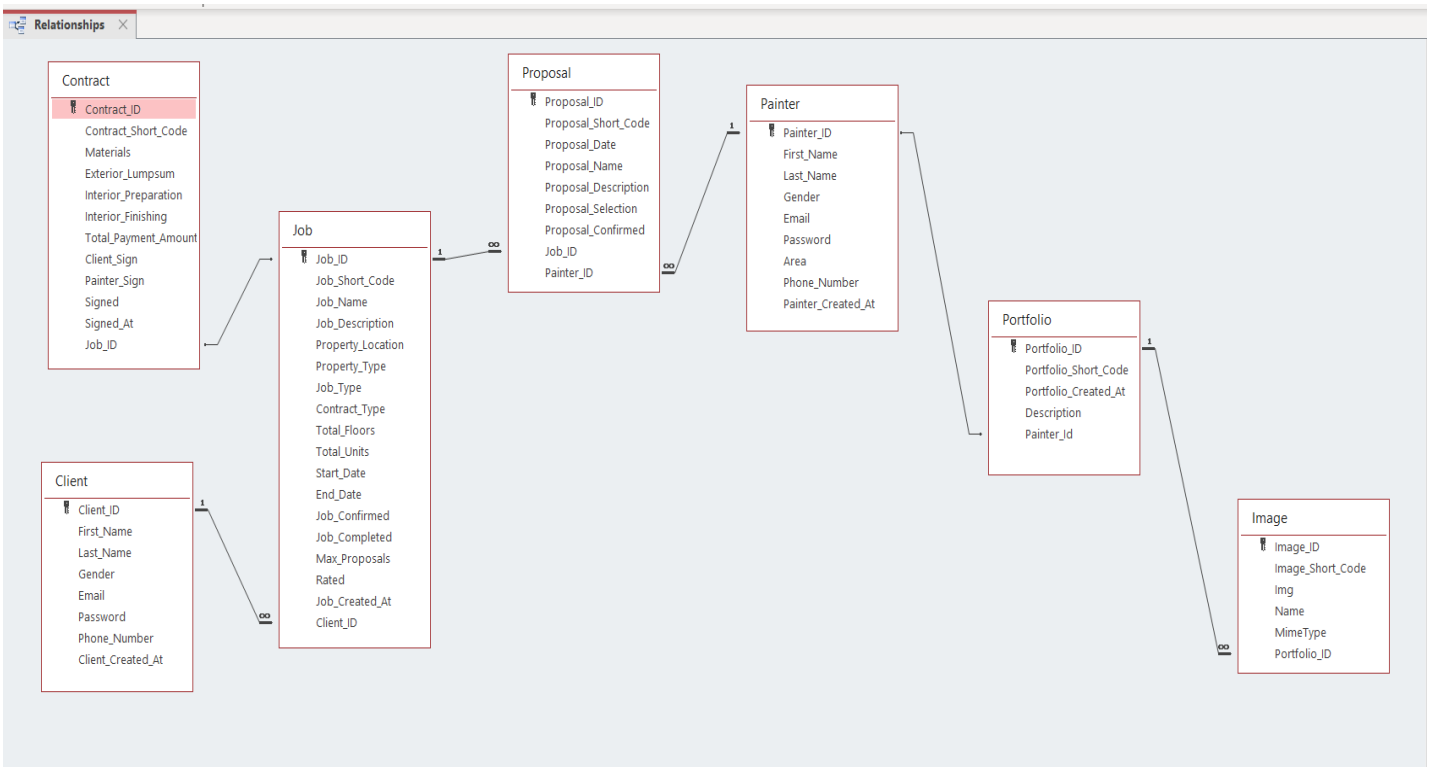
| Field | Value |
|---|---|
| Name | Rating No |
| Description | The actual rating value |
| Data Type | Integer |
| Validation | Not Null |
| Length | - |

### Painter ID

| Field | Value |
|---|---|
| Name | Painter ID |
| Description | ID of the painter associated with the rating |
| Data Type | Integer |
| Validation | - |
| Foreign Key | True |

# Database Design

## *Entity Relationship Diagram*



*Relationships:*

### Client to Job

One to Many Relationship

Creates a link between a client and a job. A client can have many jobs, but a job can belong to 1 client.

### Job to Proposal

One to Many Relationship

A job can have many proposals, but a proposal can belong to one job.

### Painter to Proposal

One to Many Relationship

A painter can have many proposals, but a proposal can belong to only 1 painter. It creates a link between a job and a painter.

### Job to Contract

One to One Relationship

A job can have only 1 contract while a contract can belong to only one job.

### Painter to Portfolio

One to One Relationship

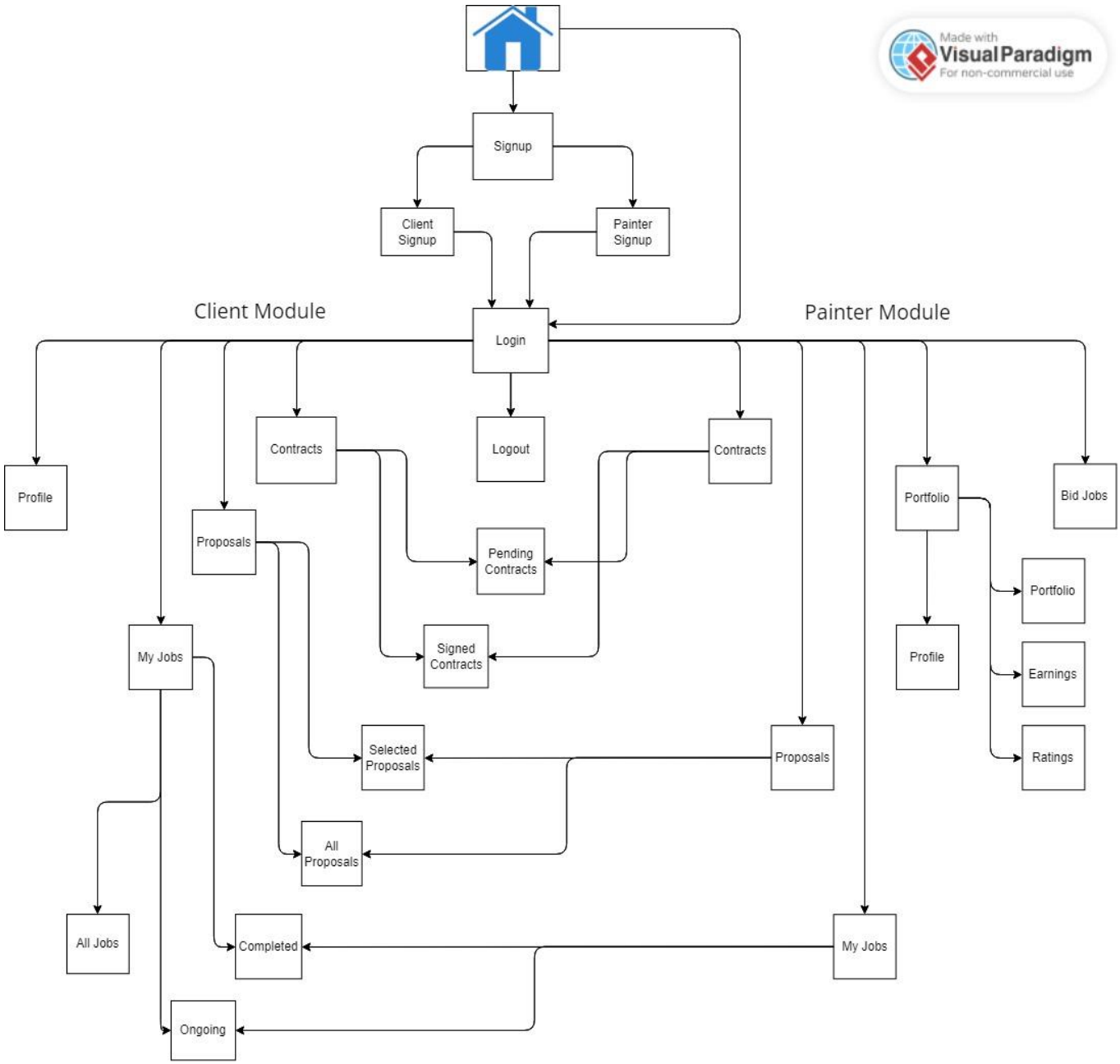
A painter can have only one portfolio while a portfolio can belong to only one painter.

### Portfolio to Image

One to Many Relationship

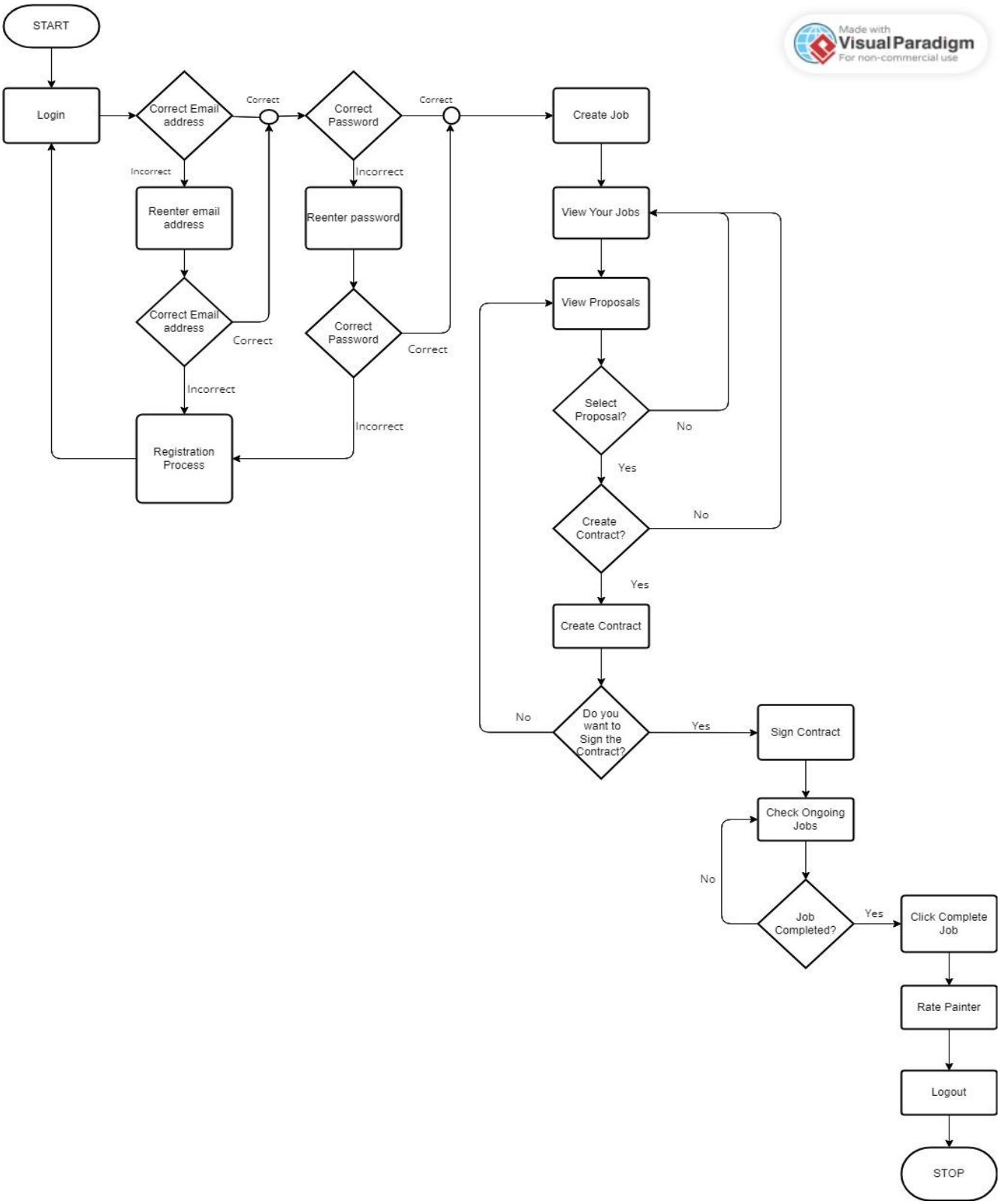
A portfolio can have several images while an image can belong to only one portfolio.

## Site Map:

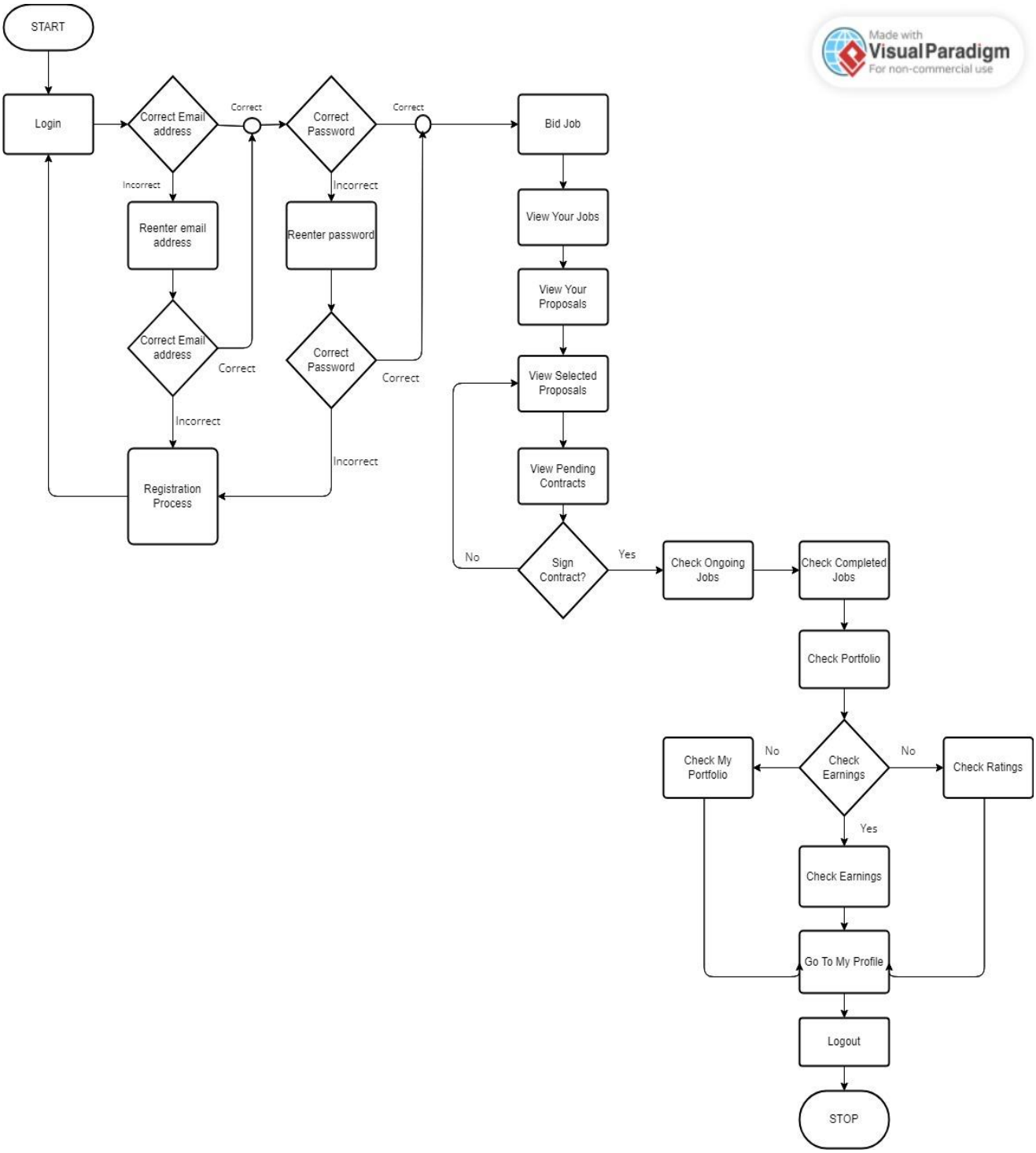

Signup
Client Signup
Painter Signup
Client Module
Painter Module
Login
Profile
Contracts
Logout
Contracts
Portfolio
Bid Jobs
Proposals
Pending Contracts
Portfolio
My Jobs
Signed Contracts
Profile
Earnings
Selected Proposals
Proposals
Ratings
All Proposals
All Jobs
Completed
My Jobs
Ongoing

## User Flows:

### *Client*



*Painter*


START
Login
Correct Email address
Correct
Correct Password
Correct
Bid Job
Incorrect
Reenter email address
Incorrect
Reenter password
View Your Jobs
View Your Proposals
Correct Email address
Correct
Correct Password
Correct
View Selected Proposals
Incorrect
Incorrect
View Pending Contracts
Registration Process
No
Sign Contract?
Yes
Check Ongoing Jobs
Check Completed Jobs
Check Portfolio
Check My Portfolio
No
Check Earnings
No
Check Ratings
Yes
Check Earnings
Go To My Profile
Logout
STOP

# SYSTEM IMPLEMENTATION

## Resources Used:

To successfully develop and deploy the Marangi Painting Management System, the following resources were required:

a) A laptop or desktop computer with adequate processing power and storage space: This is necessary to run the development tools and store project files.

b) Internet connection: An internet connection is needed to access online resources and facilitate payment processing within the system.

c) Web development tools: React is used for the frontend development, providing a JavaScript library for building user interfaces. Python Flask is used for backend development, handling server-side logic and integrating with the database.

d) Database management system (DBMS): MySQL is utilized as the DBMS for storing customer information and job details. It provides a structured and efficient way to manage and retrieve data.

e) Expertise in web development and database management: Proficiency in web development technologies like React, Flask, and MySQL is essential to effectively build and maintain the Marangi Painting Management System.

f) Experienced painter: The insights and expertise of an experienced painter are valuable in understanding the specific needs of both painters and customers, contributing to the system's functionality and user experience.

## Technologies Used:

### ReactJS:

React is used for the front-end development, enabling the creation of interactive user interfaces using JavaScript. HTML, CSS, and JSX are used to structure and style the web pages. React-icons are used for icons in the application.

### Python Flask and SQLAlchemy:

Flask is used as the backend framework, handling requests, processing data, and integrating with the database. SQLAlchemy is employed as an Object-Relational Mapping (ORM) tool for seamless interaction with the database.

### Visual Studio Code (VSCode):

VSCode is the Integrated Development Environment (IDE) used for coding. It offers a feature-rich environment and supports various programming languages.

### Insomnia:

Insomnia is used for creating APIs and testing server requests in the application. It helps in debugging and ensuring the proper functioning of the backend.

### .env file:

The .env file stores important environment variables such as the secret key, SQLALCHEMY_TRACK_MODIFICATIONS flag, and the upload folder path for images.

### Karla font:

The Karla font, downloaded from Google Fonts, is used to style the text within the application.

### Google Cloud's Geolocation API:

The Geolocation API provided by Google Cloud is utilized to retrieve people's locations and determine their time zones, enhancing the functionality of the application.

## Testing

### Frontend optimization:

To accelerate frontend development, data request issues are identified and addressed using Insomnia.

### Localhost configuration:

React uses localhost:3000, while Flask uses localhost:5000. The connection between the frontend and backend is established by adding a key-value pair in React's package.json file, specifying the proxy as "http://localhost:5000".

### User Testing:

During the testing phase, I actively engaged with construction managers and skilled painters to gather feedback and insights to improve the Marangi painter management system. Their expertise and collaboration proved invaluable in refining the application's design and functionality. By discussing the system with them, I gained a deeper understanding of their requirements and identified areas where modifications were needed.

For instance, based on their input, I added a new field called "Contract Type" to the job filling process. This field allowed users to specify whether the job required labor only or both labor and materials. This modification enhanced the system's flexibility and better aligned it with the needs of construction managers and painters.

Furthermore, through discussions and brainstorming sessions with these stakeholders, we iteratively refined the system's design, making it more intuitive and user-friendly. Their feedback helped identify usability issues, which we addressed by making necessary adjustments and improvements to the user interface.

Their active involvement in the testing process also played a significant role in identifying and resolving bugs or functional issues. They provided valuable bug reports, allowing me to debug and fix issues promptly.

Overall, the collaboration with construction managers and skilled painters greatly contributed to the success of the Marangi painter management system. Their feedback, suggestions, and willingness to participate in the testing process helped shape the application into a more effective and efficient tool for managing painting projects.

# CONCLUSION

The Marangi Painting Management System aims to address the challenges faced by both customers and painters in the current manual system. By providing a centralized platform for customers to easily find skilled painters and for painters to access potential job opportunities, the system streamlines the process of connecting clients with suitable painters.

The system enables customers to register, post their painting jobs or projects, and allocate them to painters based on proximity and suitability. This eliminates the need for customers to rely solely on word-of-mouth recommendations and expands their choices of painters. Additionally, the system allows painters to showcase their portfolios, providing customers with a concrete understanding of the painters' skills and previous work.

Painters also benefit from the system by gaining access to a wider pool of job opportunities. Instead of relying solely on referrals or scouting construction sites, painters can actively search for available jobs through the platform. The system facilitates efficient job management, allowing painters to track progress, meet deadlines, and communicate updates to clients.

Furthermore, the system addresses the manual and paper-based record-keeping practices currently employed by painters. By digitizing data storage and providing a secure database, the system eliminates the risk of data loss and allows for efficient management of customer information and financial transactions. The system also enables painters to generate contracts, invoices, and track revenue, providing them with a comprehensive toolset for managing their business operations.

Overall, the Marangi Painting Management System revolutionizes the way customers and painters interact and conduct business in the painting industry. By leveraging technology, the system enhances the customer experience, expands opportunities for painters, and improves the efficiency and effectiveness of job management and communication between both parties.

# References

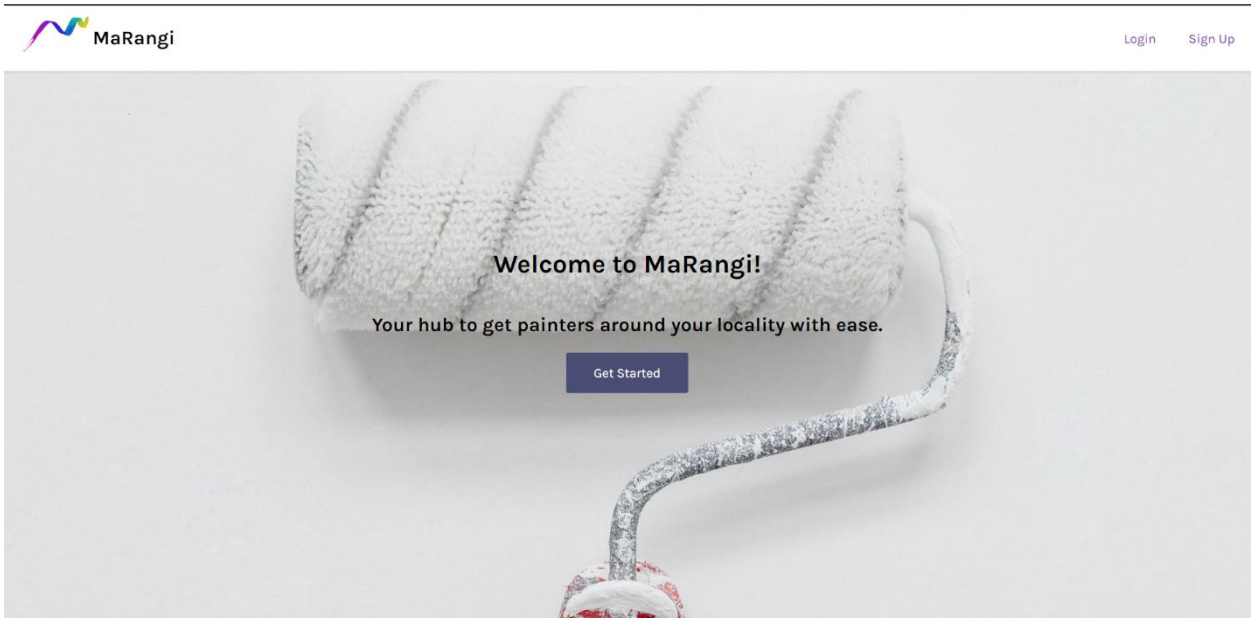
Jiji - “Https://Jiji.co.ke/Nairobi-Central/Building-And-Trades-Services/Interior-And-Exterior-House-Painting-Services-UPtRUx6pU32uv7P87GRuOazx.html.” *Jiji.co.ke*, jiji.co.ke/nairobi-central/building-and-trades-services/interior-and-exterior-house-painting-services-uPtRUx6pU32uv7P87GRuOazx.html?page=1&pos=2&cur_pos=2&ads_per_page=19&ads_count=171&lid=LnZQ-L_VYUepWGMk&indexPosition=1. Accessed 27 Feb. 2023.

Paints, Crown. “Find a Painter - Homepage.” *Findapainter.crownpaints.co.ke*, 2022, findapainter.crownpaints.co.ke/. Accessed 27 Feb. 2023.

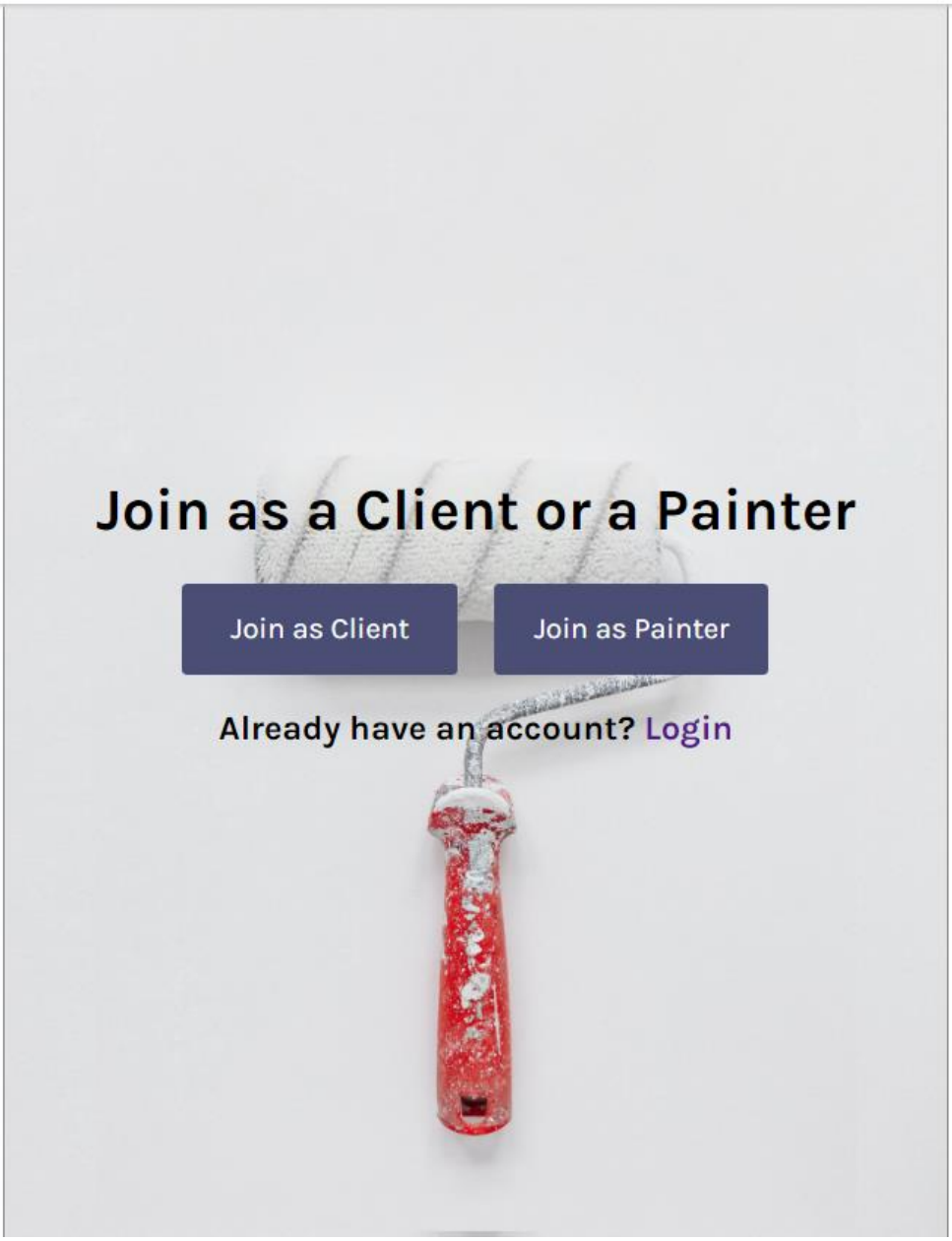
# APPENDIX A: USER MANUAL

## Home Pages

I) Home Page

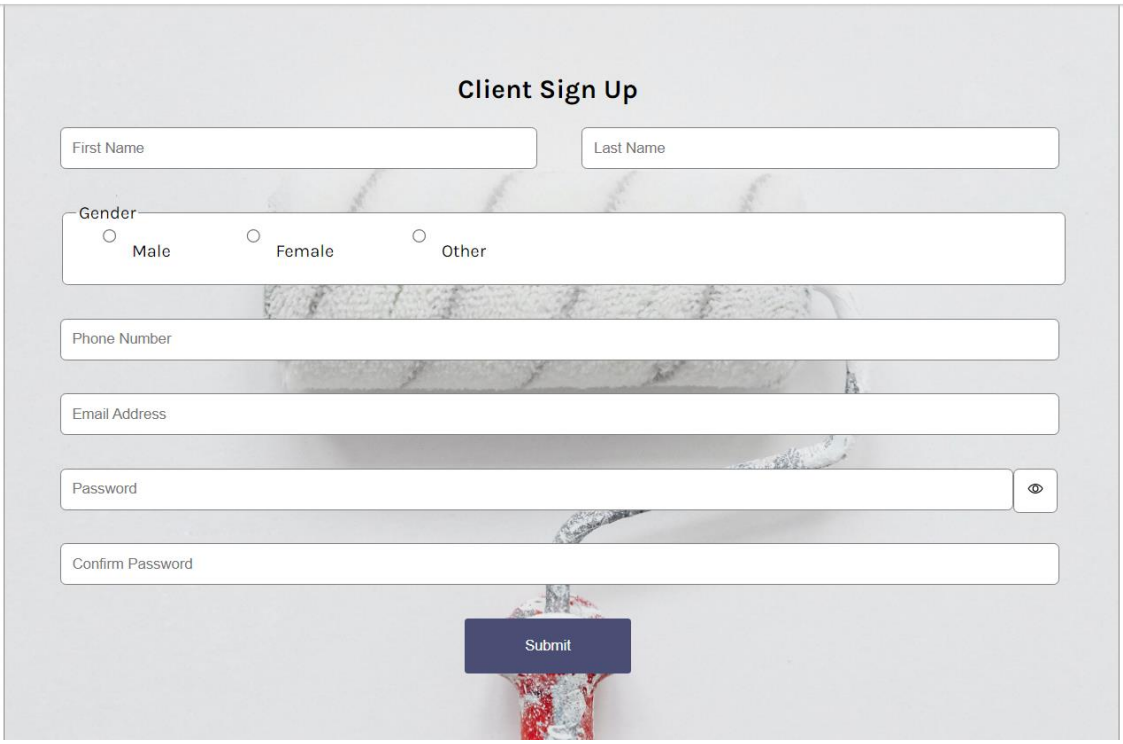


II) Join as a Client or Painter

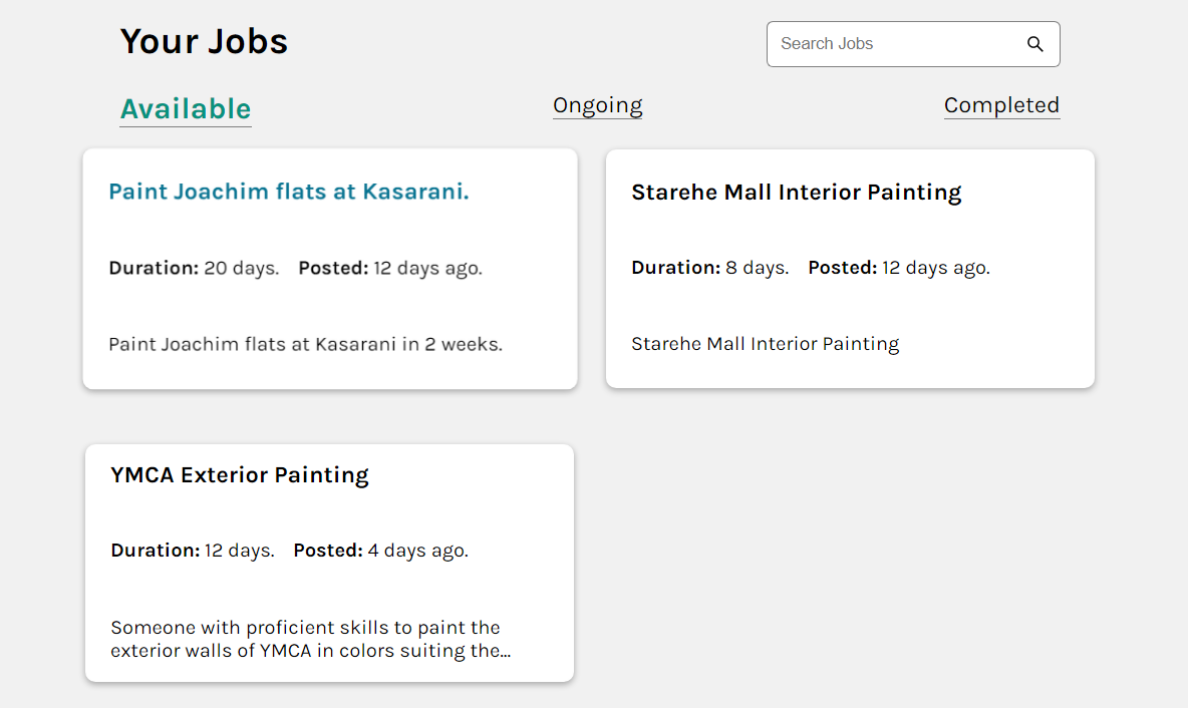


## Client

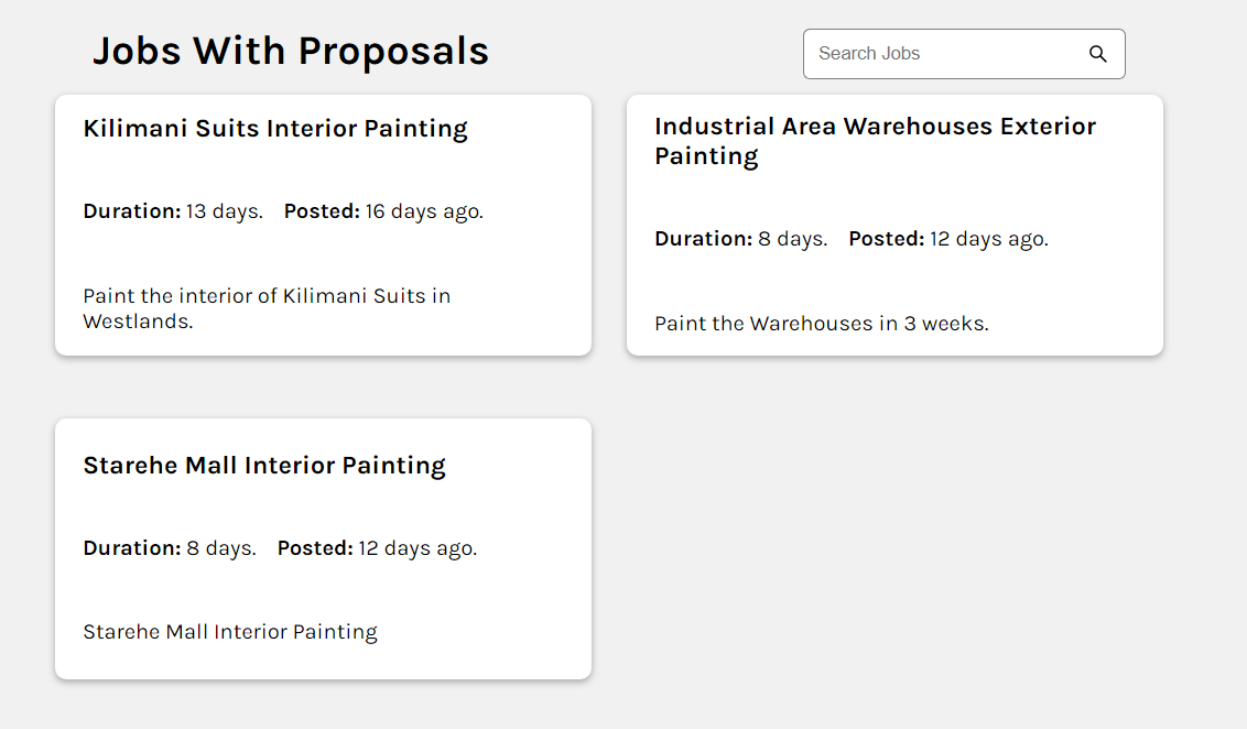
i) Client Signup



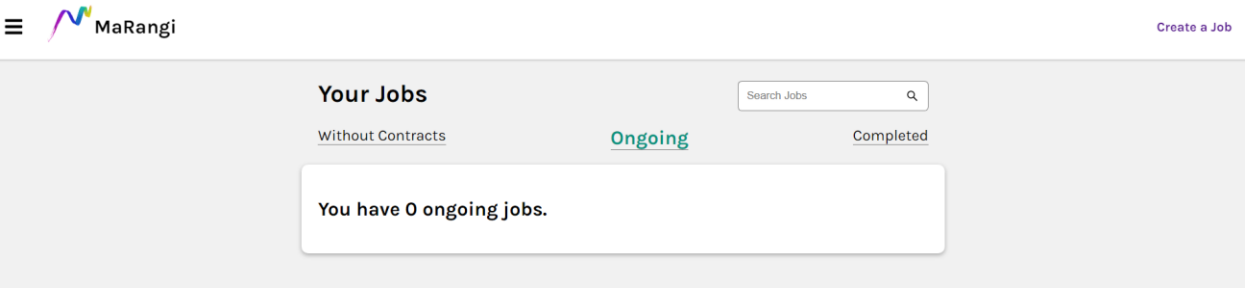

ii) Client Jobs



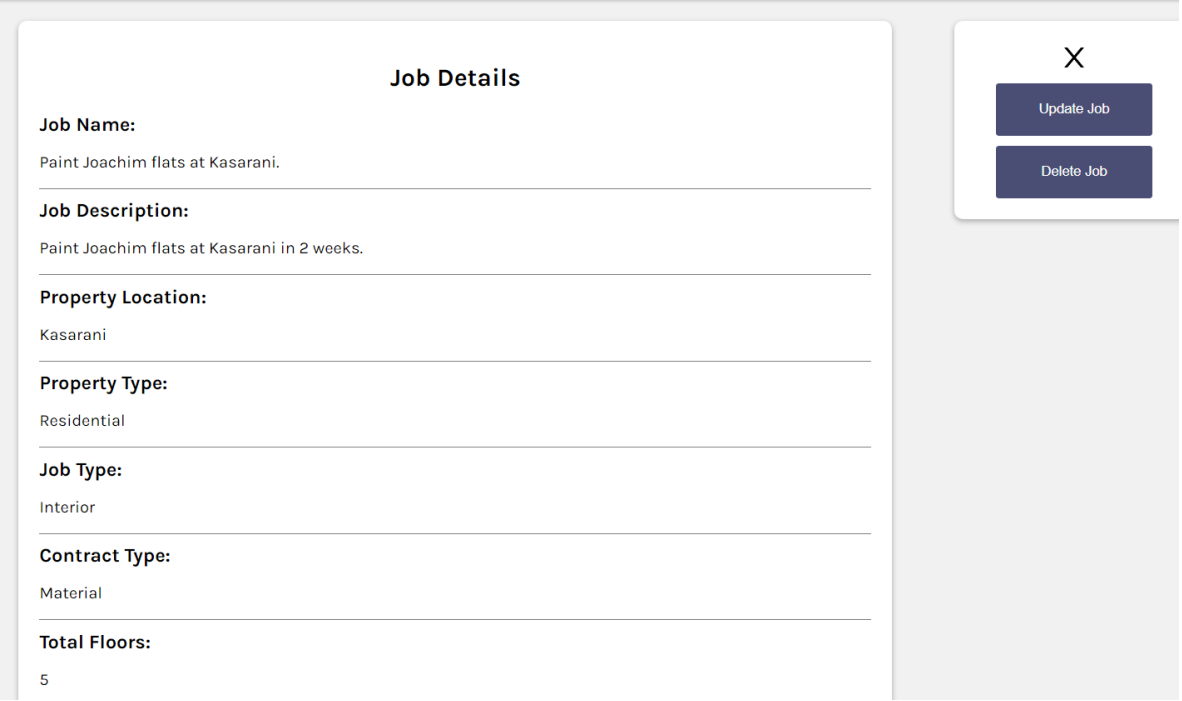
iii) Jobs with Proposals




iv) Ongoing Jobs

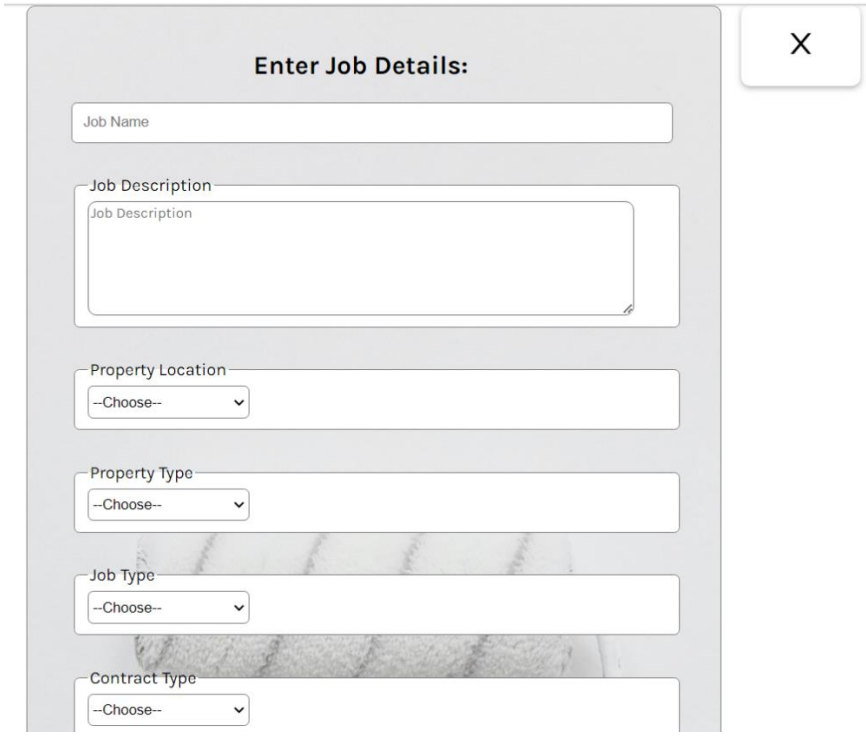


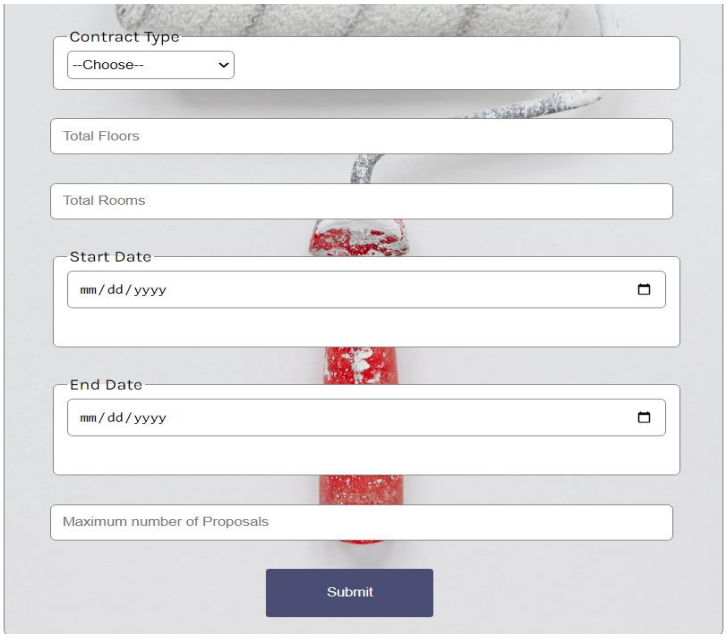
v) Job Details



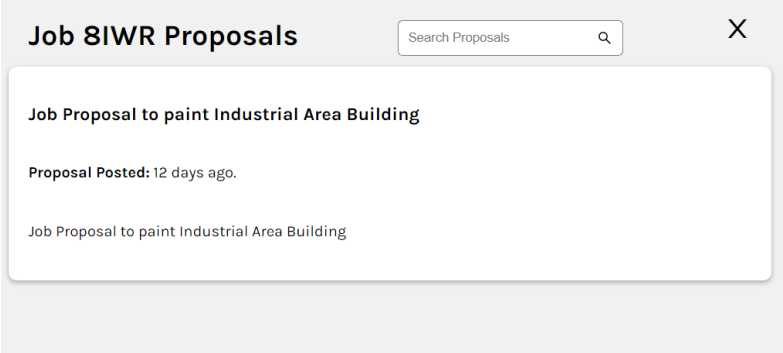

vi) Delete Job

viii) Create a Job

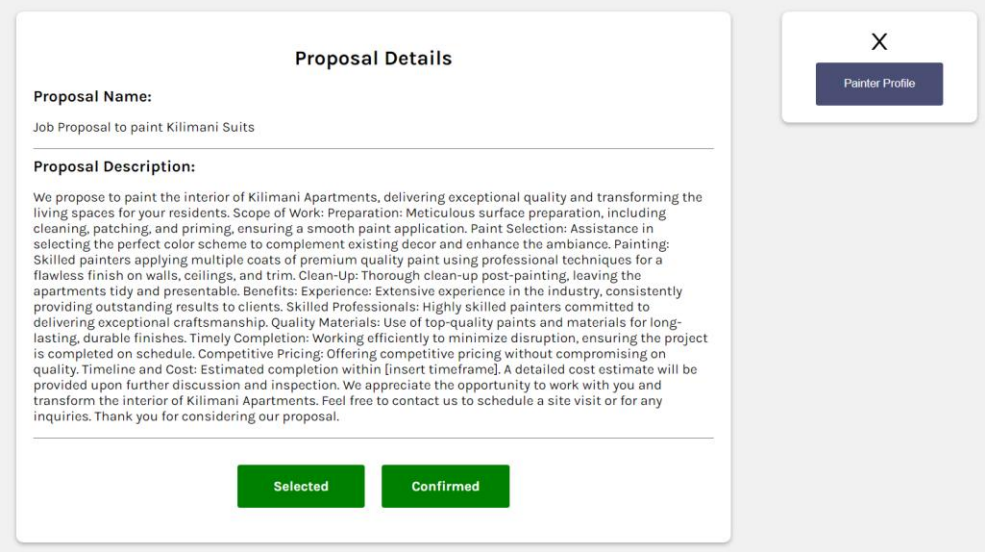


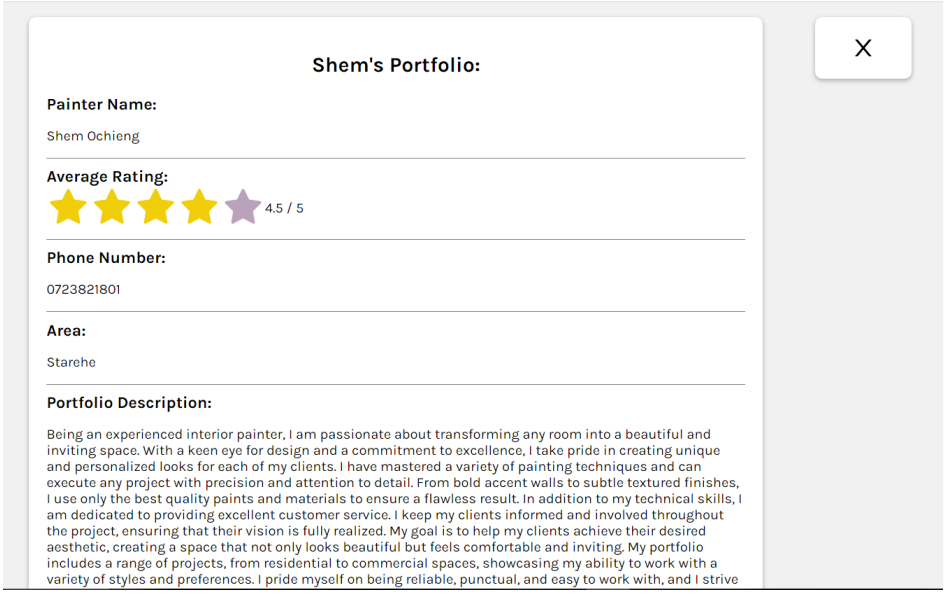
ix) Proposals for a Job:



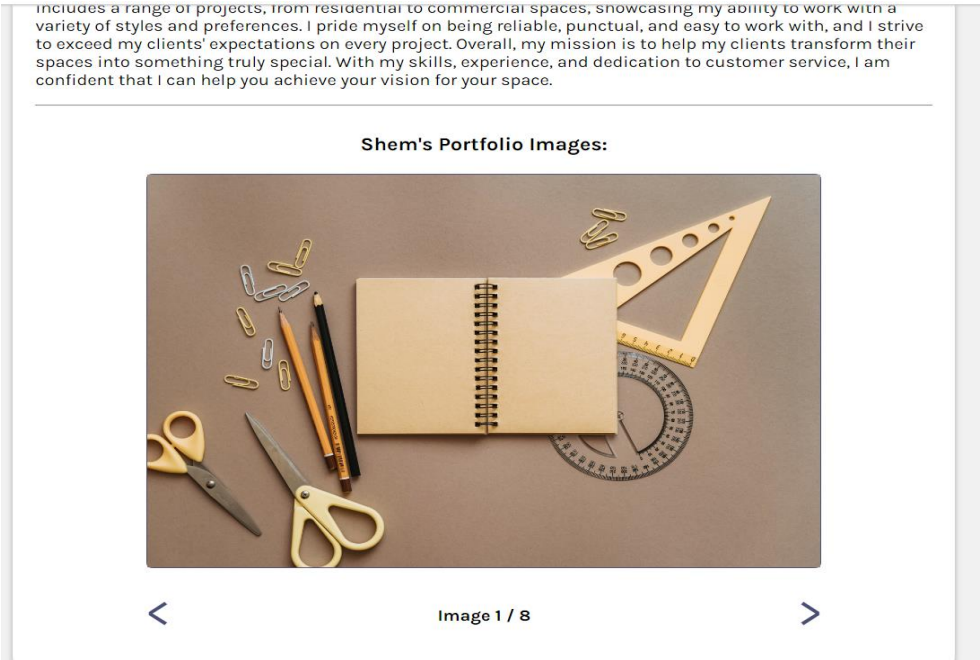

x) Selected Proposal



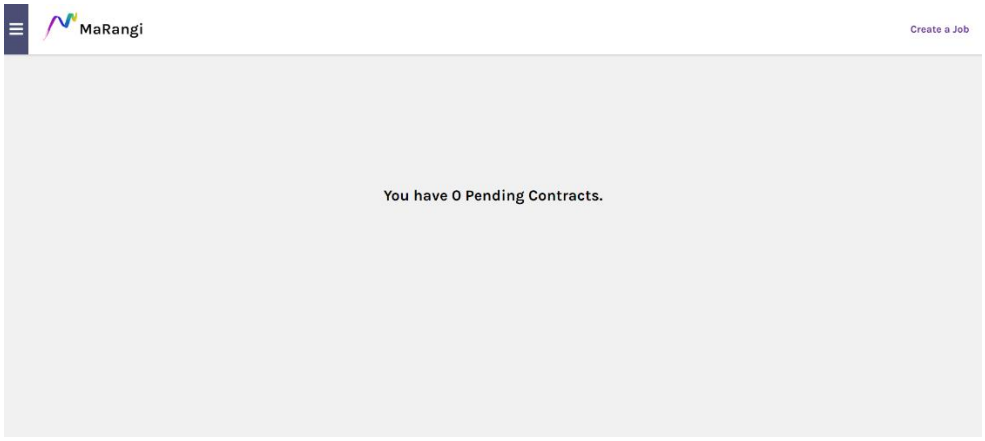

xi) Painter Portfolio



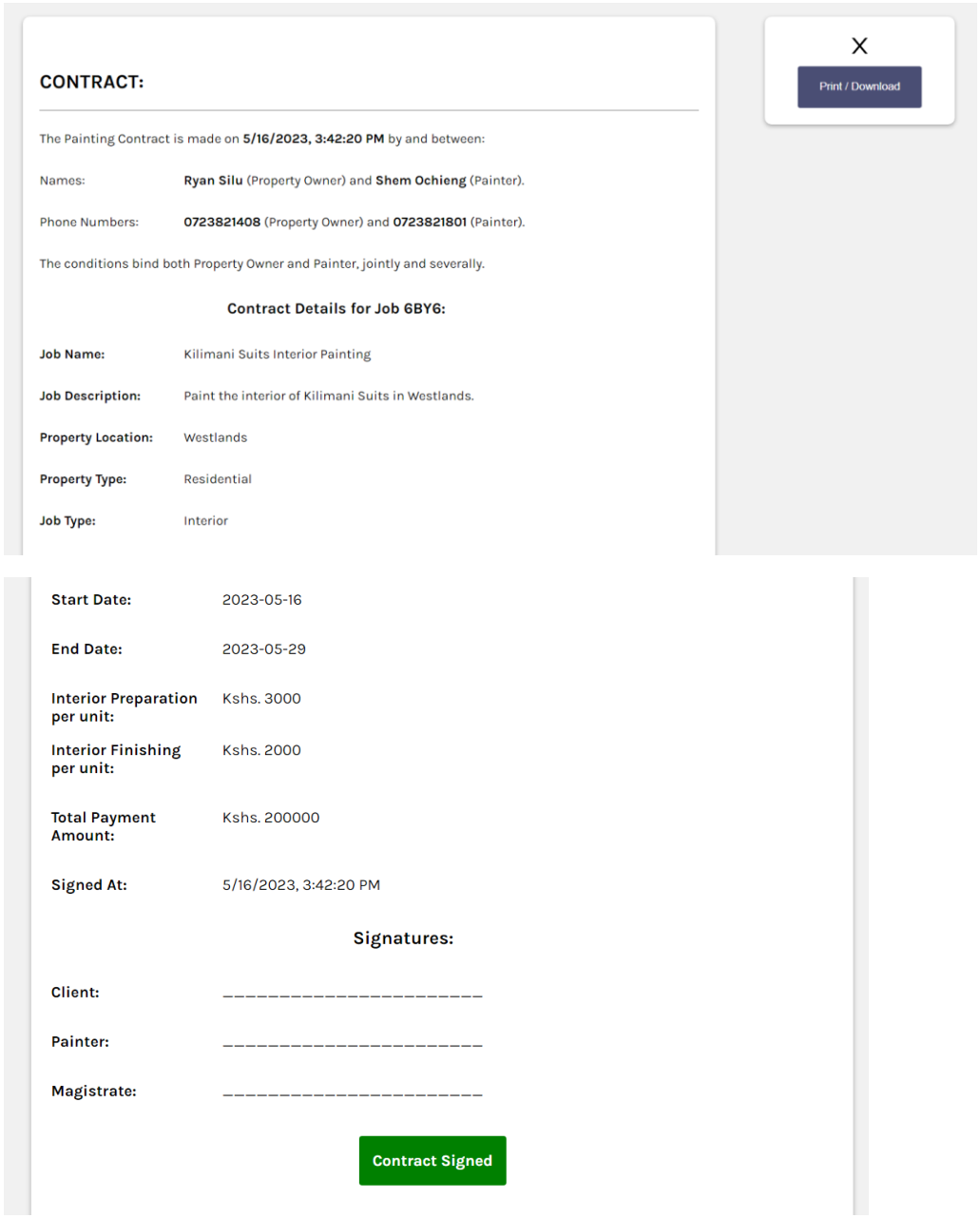
xii) Select Portfolio Images




xiii) Pending Contract



xiv) Contract



CONTRACT:

The Painting Contract is made on **5/16/2023, 3:42:20 PM** by and between:

Names: **Ryan Silu** (Property Owner) and **Shem Ochieng** (Painter).

Phone Numbers: **0723821408** (Property Owner) and **0723821801** (Painter).

The conditions bind both Property Owner and Painter, jointly and severally.

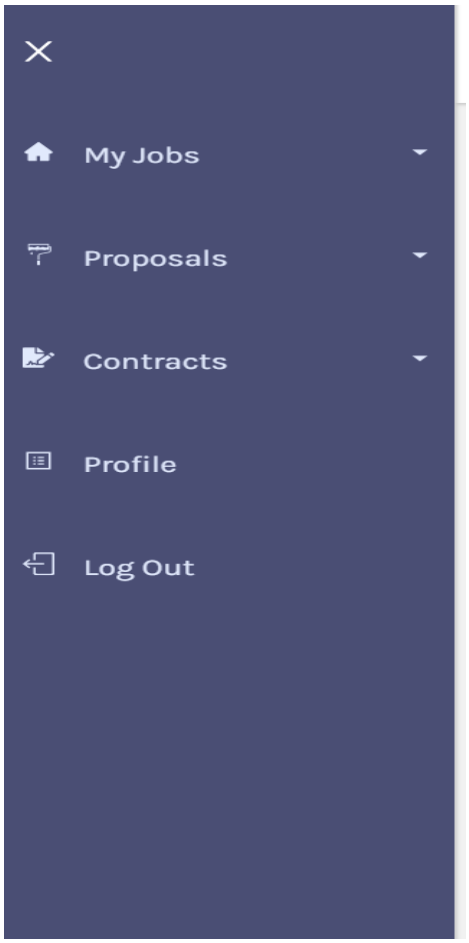
**Contract Details for Job 6BY6:**

| | |
|---|---|
| **Job Name:** | Kilimani Suits Interior Painting |
| **Job Description:** | Paint the interior of Kilimani Suits in Westlands. |
| **Property Location:** | Westlands |
| **Property Type:** | Residential |
| **Job Type:** | Interior |
| **Start Date:** | 2023-05-16 |
| **End Date:** | 2023-05-29 |
| **Interior Preparation per unit:** | Kshs. 3000 |
| **Interior Finishing per unit:** | Kshs. 2000 |
| **Total Payment Amount:** | Kshs. 200000 |
| **Signed At:** | 5/16/2023, 3:42:20 PM |

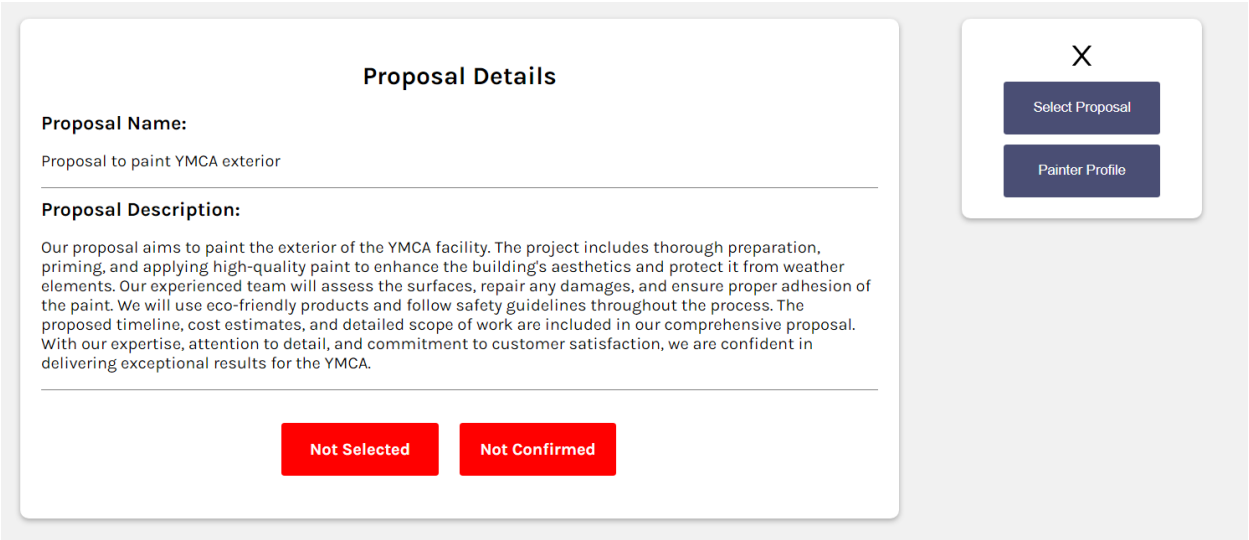
**Signatures:**

| | |
|---|---|
| **Client:** | _____ |
| **Painter:** | _____ |
| **Magistrate:** | _____ |

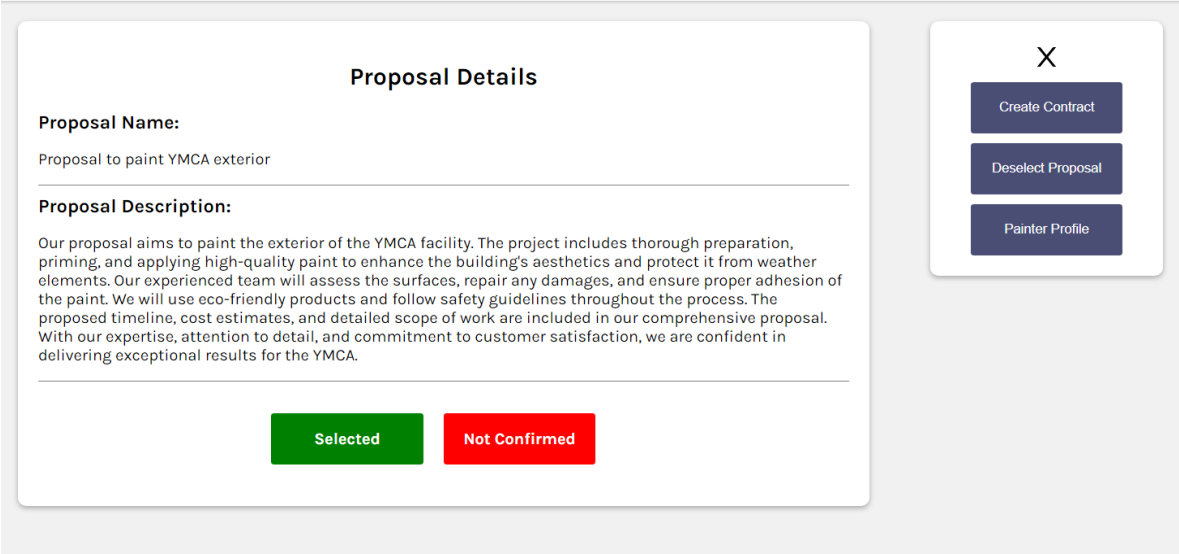
xv) Client Sidebar



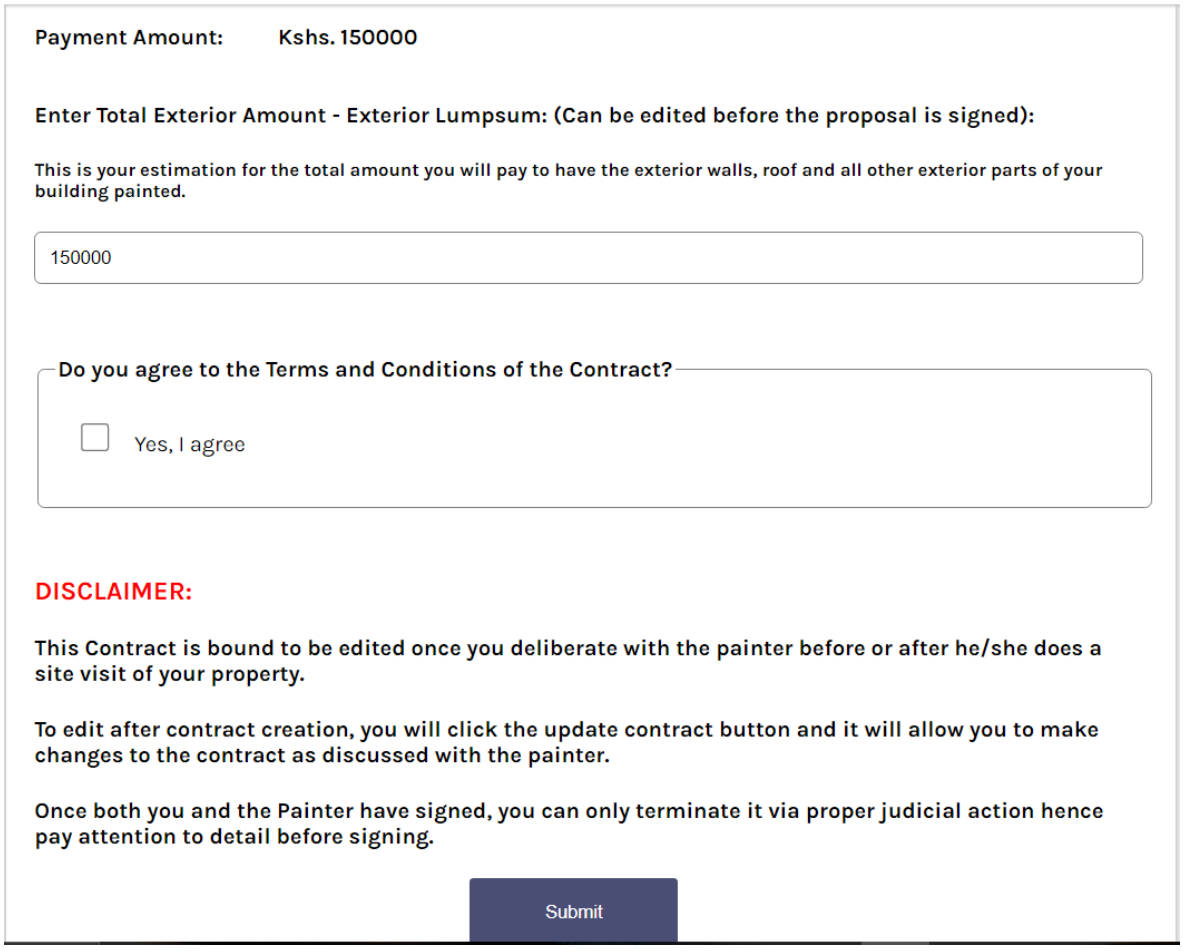

xvi) Select Proposal



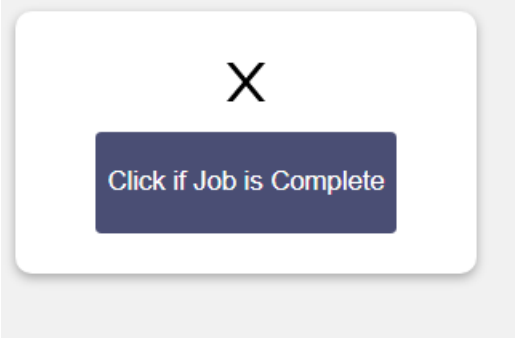
xvii) Create Contract Button



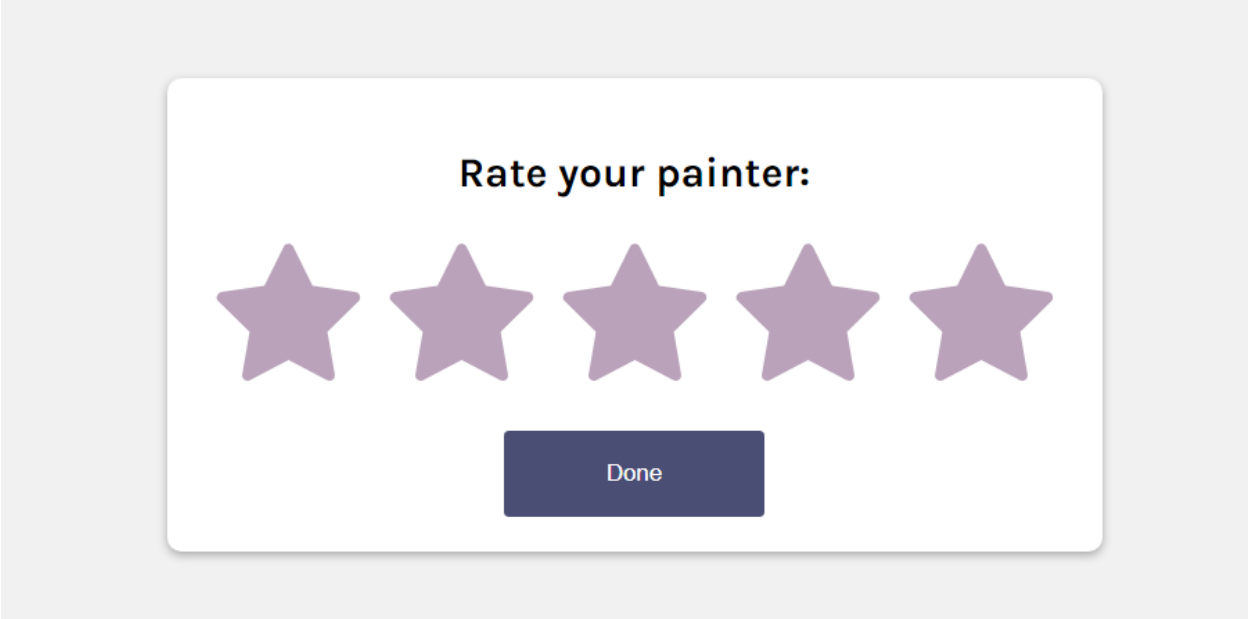

xviii) Create Contract



xix) Complete Job:

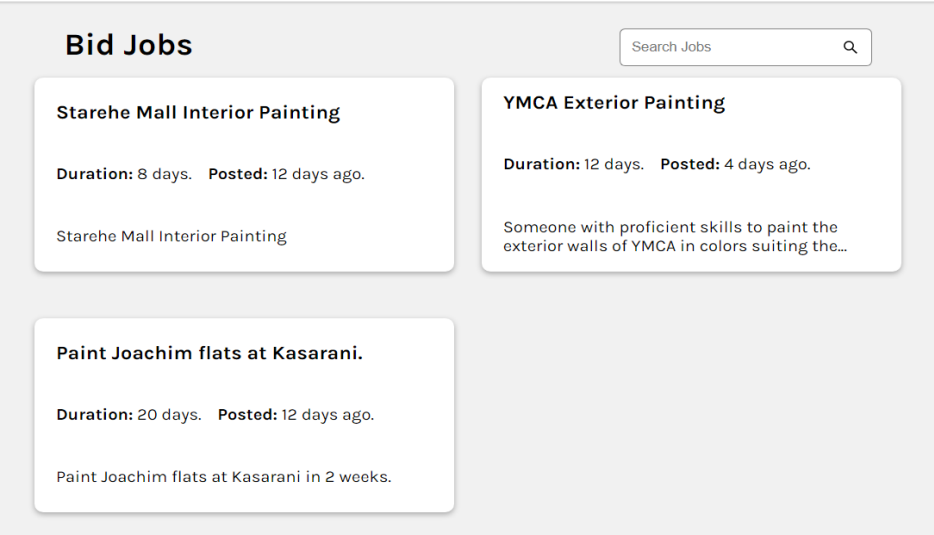



xx) Rate Painter

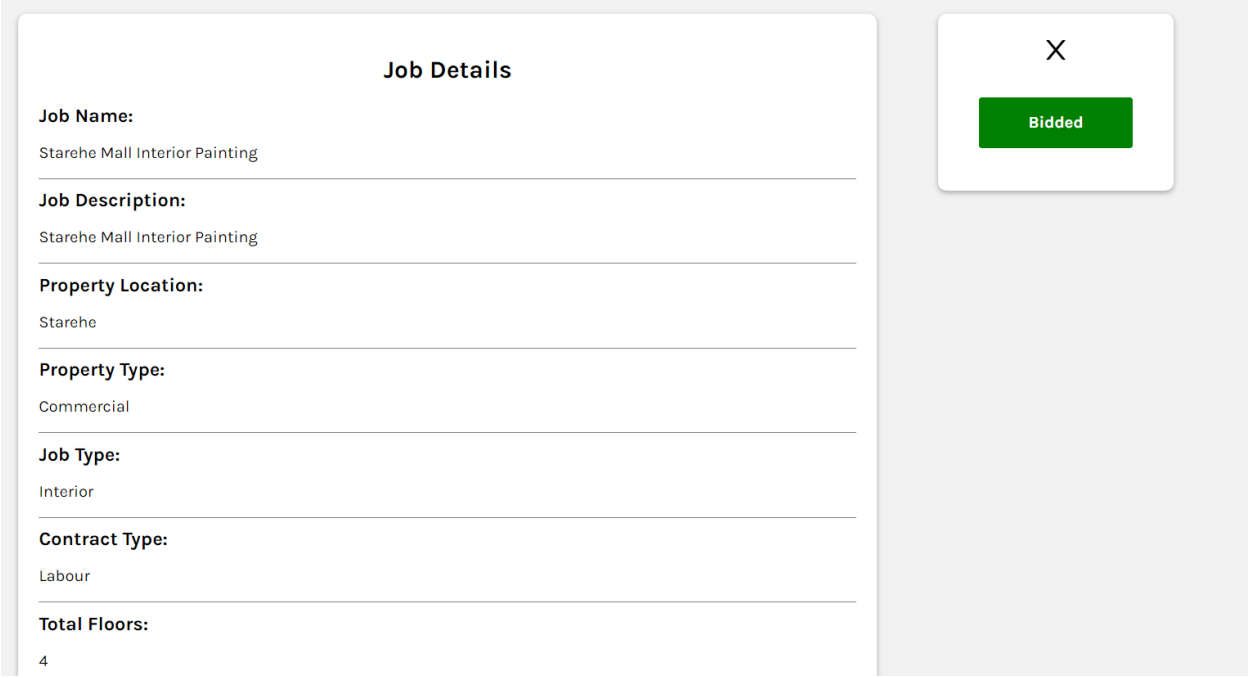


Painter:

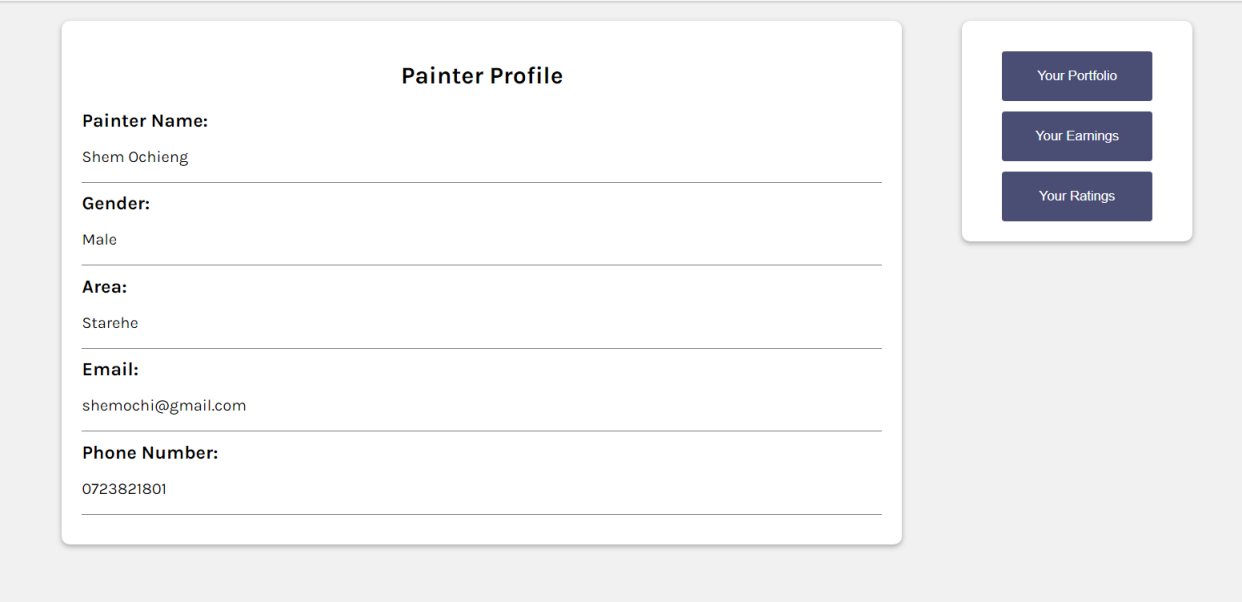
I) Bid Jobs



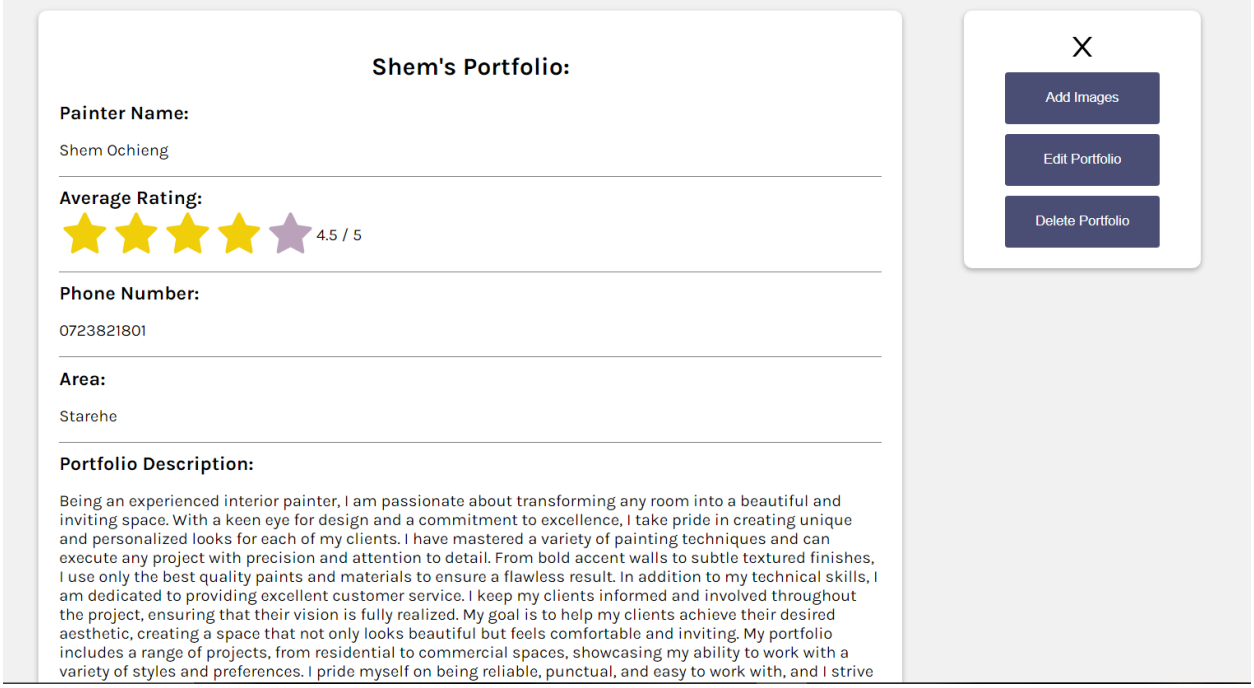

II) Specific Job



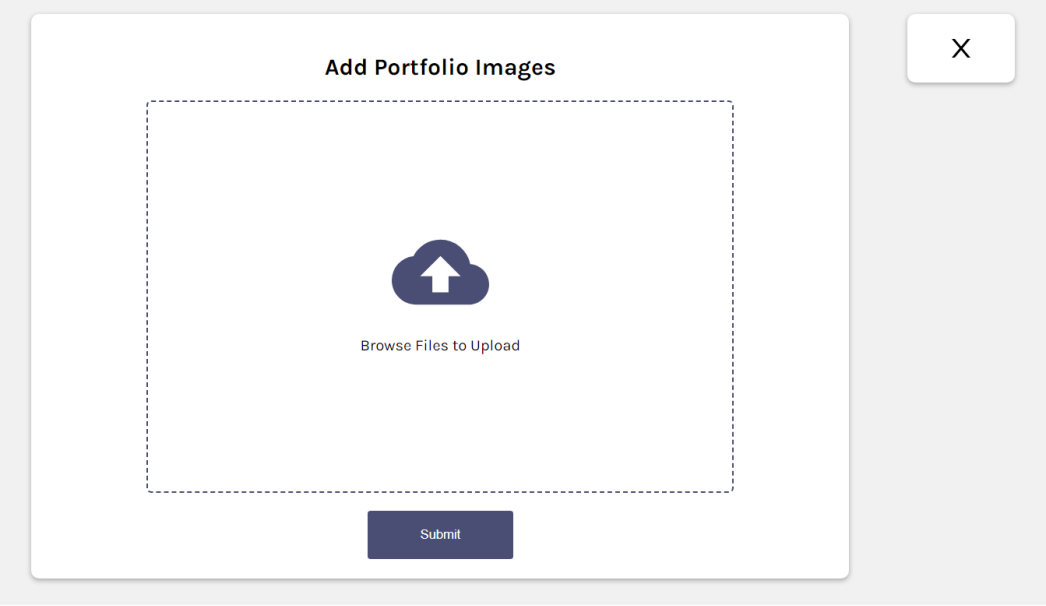
III) Painter Profile

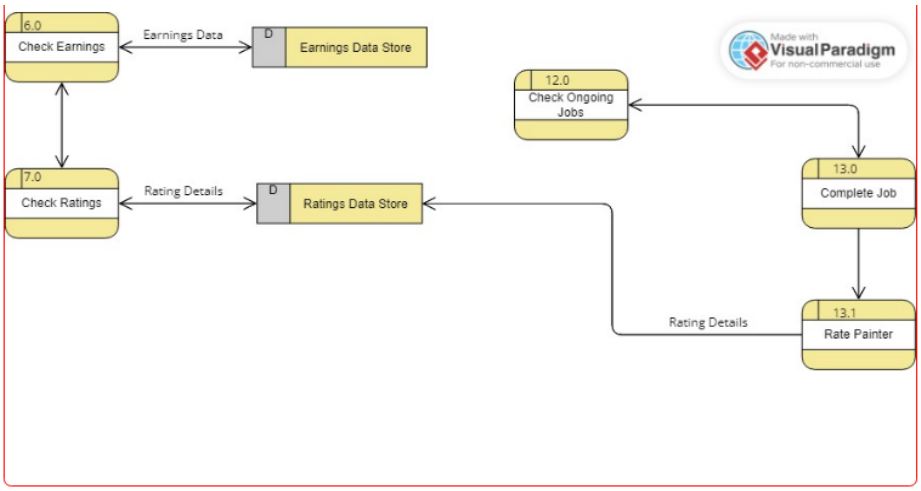



IV) Painter Portfolio



V) Add Images:




Cannot have more than 10 files in your portfolio!

Submit

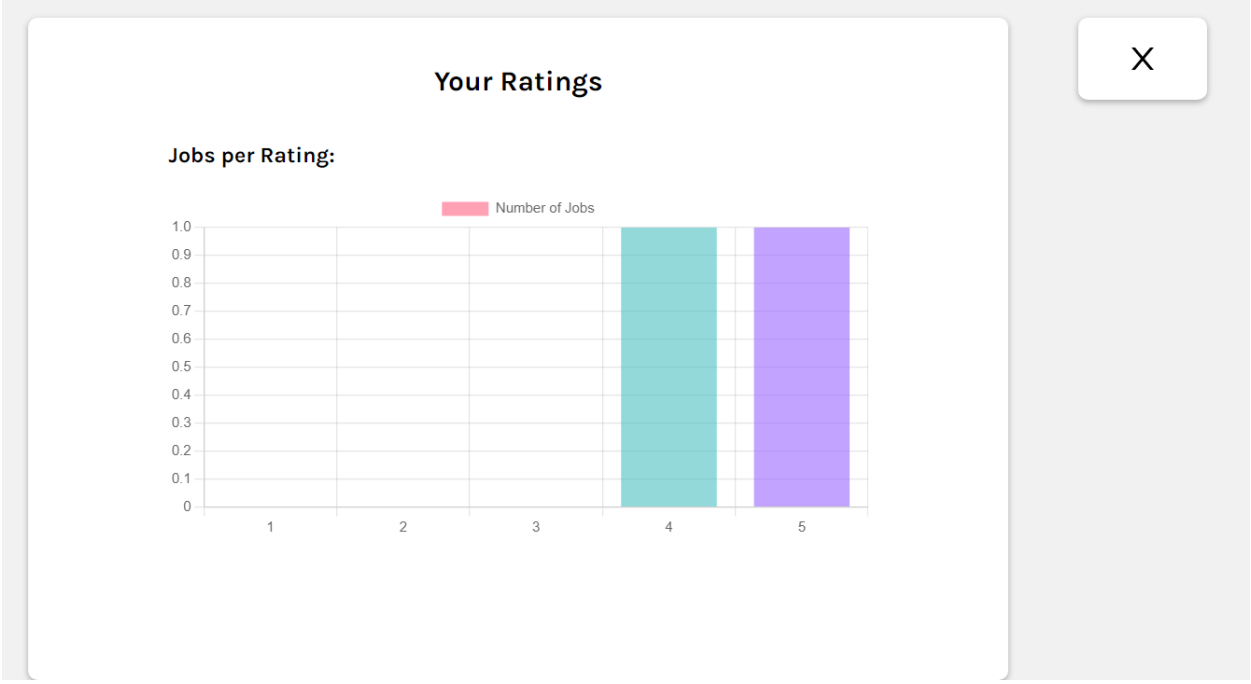
DFD (7).jpg

DFD (6).jpg

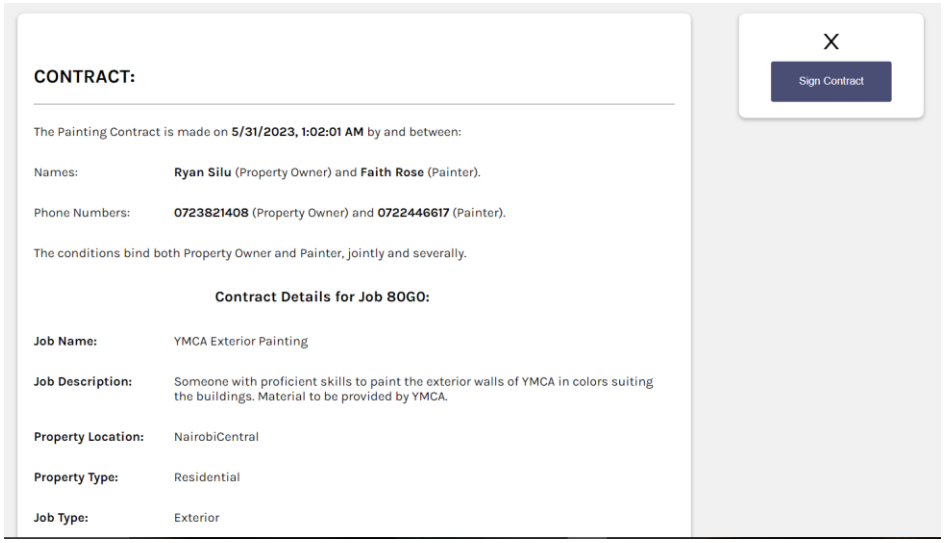
DFD.vpd (14).jpg

VI) Earnings:

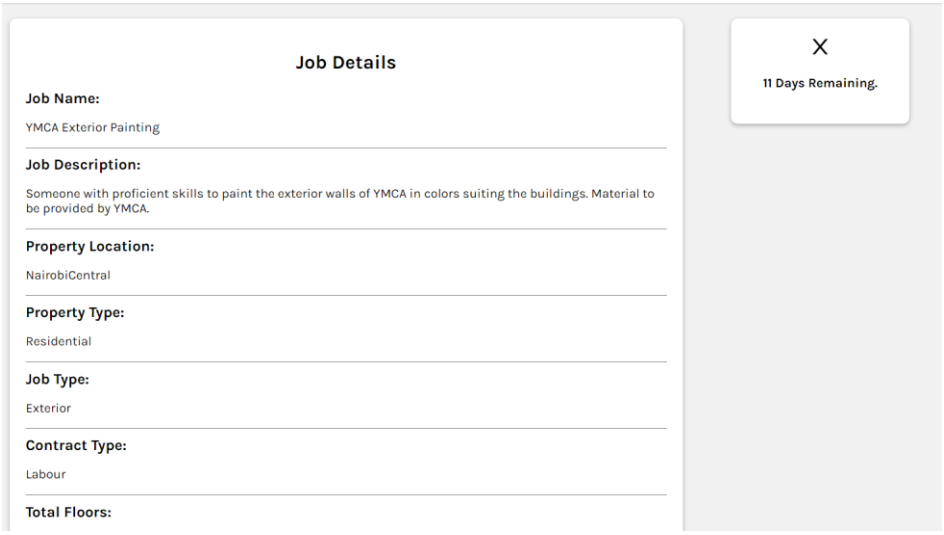

VII) Ratings



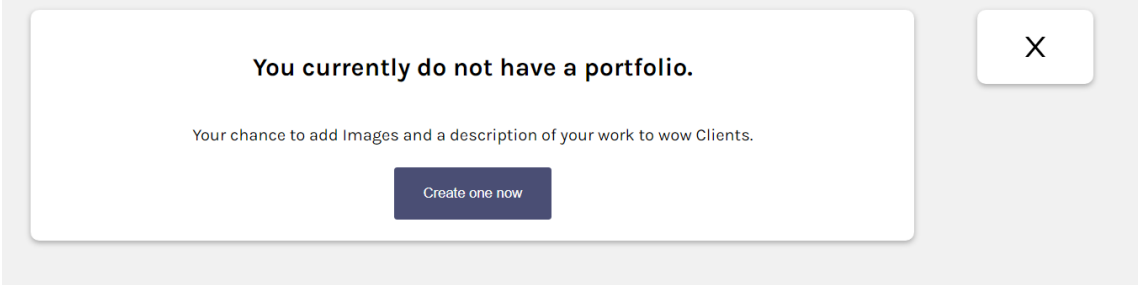
VIII) Sign Contract




IX) Confirmed Job Details




X) Create Portfolio



# APPENDIX B: Sample Code

## Python Flask

I) Contract Model

```
class Contract(db.Model):
    __tablename__= "contract"
    id = db.Column(db.Integer(), primary_key = True)
    contract_short_code = db.Column(db.String(), nullable = False, unique = True)
    materials = db.Column(db.String(), nullable = True)
    exterior_lumpsum = db.Column(db.Integer(), nullable=True)
    interior_preparation = db.Column(db.Integer(), nullable=True)
    interior_finishing = db.Column(db.Integer(), nullable=True)
    total_payment_amount = db.Column(db.Integer(), nullable=False)
    client_sign = db.Column(db.Boolean(), default = False, nullable = False)
    painter_sign = db.Column(db.Boolean(), default = False, nullable = False)
    signed = db.Column(db.Boolean(), default = False, nullable = False)
    signed_at = db.Column(db.DateTime(), nullable = True)
    job_id = db.Column(db.Integer(), db.ForeignKey("job.id"))

    def __repr__(self):
        return f"<Contract {self.id}>"

    def save(self):
        db.session.add(self)
        db.session.commit()

    def delete(self):
        db.session.delete(self)
        db.session.commit()
```

Creates the contract module in the database.

II) Job Creation

```
    job_created_at=datetime.now(local_tz)
    new_job = Job (
        job_short_code = code,
        job_name = data.get("job_name"),
        job_description = data.get("job_description"),
        property_location = data.get("property_location"),
        property_type = data.get("property_type"),
        job_type = data.get("job_type"),
        contract_type = data.get("contract_type"),
        total_floors = data.get("total_floors"),
        total_rooms = data.get("total_rooms"),
        start_date = start_date,
        end_date = end_date,
        max_proposals = data.get("max_proposals"),
        job_created_at = job_created_at,
        client_id = current_client.id
    )

    new_job.save()
    response = make_response(jsonify({"message": "Job created successfully!"}))
    response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    return response
```

Saves a new job.

III) Running the backend

```python
from app import create_app
from config import DevConfig

app = create_app(DevConfig)

if __name__ == "__main__":
    app.run()
```

## React JS

I) Return the whole app:

```
    return (
        <div>
            <CredentialsApp2 />
            {/* {message} */}
        </div>
    )
}
```

II) Input Painter's First and Last Name

```
<h2 className="form-title">Painter Sign Up</h2>
<div className="names-pass">
    <div className = "name1">
        <input
            id="first_name"
            type="text"
            placeholder="First Name"
            name="first_name"
            value = {painterForm.first_name}
            onChange={updateValues}
            className = "name"
            style = {(submittedEmpty.includes("first_name")) ? styles : null}
        />
    </div>
    <div className = "name2">
        <input
            id="last_name"
            type="text"
            placeholder="Last Name"
            name="last_name"
            value = {painterForm.last_name}
            onChange={updateValues}
            className = "name"
            style = {(submittedEmpty.includes("last_name") || !lenFName || !lenLName) ? styles : null}
        />
    </div>
```

Receives the values of first name and last name and stores them in a state called Painter-Form that has all attributes to be input by the painter during registration.

III) Rating a Painter – Rendered Stars

```
    let count = 5
    const starRating = useMemo(() => {
        return Array(props.ratingProps.count)
            .fill(0)
            .map((_, i) => i + 1)
            .map(idx => {
                return (
                    <AiIcons.AiFillStar
                        key={idx}
                        className = "stars"
                        style = {getColor(idx)}
                        onClick={() => props.onRating(idx)}
                        onMouseEnter={() => setHoverRating(idx)}
                        onMouseLeave = {() => setHoverRating(0)}
                    />

                )
            })
    }, [props.ratingProps.count, props.rating, hoverRating])
    return (
        <div>
            {starRating}
        </div>
    )
}
```

It initially renders stars that have a light purple background color. Once the client chooses which star to rate the painter, the star, and each star before it, gets a golden background color.

# APPENDIX C: Sample Contract

**CONTRACT:**

The Painting Contract is made on **5/16/2023, 3:42:20 PM** by and between:

Names: **Ryan Silu** (Property Owner) and **Shem Ochieng** (Painter).

Phone Numbers: **0723821408** (Property Owner) and **0723821801** (Painter).

The conditions bind both Property Owner and Painter, jointly and severally.

**Contract Details for Job 6BY6:**

| | |
|---|---|
| **Job Name:** | Kilimani Suits Interior Painting |
| **Job Description:** | Paint the interior of Kilimani Suits in Westlands. |
| **Property Location:** | Westlands |
| **Property Type:** | Residential |
| **Job Type:** | Interior |
| **Contract Type:** | Labour |
| **Total Rooms:** | 40 Units |
| **Total Floors:** | 8 |
| **Start Date:** | 2023-05-16 |
| **End Date:** | 2023-05-29 |
| **Interior Preparation per unit:** | Kshs. 3000 |
| **Interior Finishing per unit:** | Kshs. 2000 |
| **Total Payment Amount:** | Kshs. 200000 |
| **Signed At:** | 5/16/2023, 3:42:20 PM |

**Signatures:**

| | |
|---|---|
| **Client:** | _____ |
| **Painter:** | _____ |
| **Magistrate:** | _____ |

# Annex:

## Interview Guide:

### Painter Interview Guide:

1. What tasks do you find most time-consuming or tedious in your current process for finding painting jobs and connecting with clients?
2. Which of the following marketing methods do you currently use to promote your painting services? (Select those that apply)
    a. Word-of-mouth referrals
    b. Social media accounts
    c. Business cards or flyers
    d. Website
    e. Other (please specify)
3. Have you ever used any automation tools or software to help you with marketing, client communication, or other aspects of your painting business? If so, which ones have you used and what did you like or dislike about them?
4. What challenges have you faced when communicating with potential clients or managing painting projects in the past? For example, have you ever struggled to keep track of project details, missed deadlines, or had difficulty getting in touch with clients?
5. Would you be interested in using an automated platform that helps you find painting jobs and manage client communication and projects more easily? If so, what features would be most important to you in such a platform?
6. Which of the following features would be most important to you in an automated platform for your painting business? (Select those that apply)
    a. Lead generation (finding potential clients)
    b. Client communication (for example automated emails, text messages)
    c. Project management tools (for example timelines, budgets, task lists)
    d. Invoicing and payment processing
    e. Other (please specify)
7. Do you have any concerns or reservations about using automated platforms or tools for your painting business? If so, what are they?
8. How much would you be willing to pay per month for a platform that provides the features you need to find painting jobs and manage projects more efficiently? (Select one)
    a. Kshs. 0 – 1000
    b. Kshs. 1001 - 2500

   c. Kshs. 2501 - 5000
   d. Kshs. 5001 - 10000
   e. Kshs. 10001 or more
9. Is there anything else you would like to add about your experience as a painter and the challenges you face in connecting with clients and finding painting jobs?

## Customer Interview Guide:

1. Can you describe your experience with finding and hiring painters for your projects in the past? What were some of the challenges you faced in this process?
2. What are some of the most important criteria that you look for when choosing a painter for your project? For example, do you prioritize experience, price, quality of work, or other factors?
3. If you could imagine an ideal website that would help you connect with painters who meet your criteria and preferences, what features would be most important to you? For example, would you want to be able to browse painter portfolios, read reviews from previous clients, or compare quotes from multiple painters?
4. How important is it for you to have transparency and communication throughout the painting process? Would you want to be able to communicate with your chosen painter via the website, or would you prefer to communicate via phone or email?
5. Would you be willing to pay a fee to use a website that connects you with painters, or would you expect the service to be free? (Select one)
   a. I would be willing to pay a fee.
   b. I would expect the service to be free.
6. How frequently do you anticipate needing to hire painters for your projects? Would you want to be able to save and refer to previous projects and painters on the website, or would each project be separate?
7. Are there any other features or functions that you would want to see on a website that connects you with painters for your projects?
8. Have you used any similar websites or services in the past to find and hire painters? If so, what did you like or dislike about them?
9. Is there anything else you would like to add about your experience as a client looking to hire painters and your vision for a website that could make this process easier and more efficient? (Open-ended)