**ITSP300 – A-Ford-Able Management System**

**Cover page**

|  |  |
| --- | --- |
| **Group number and name** | Group 1: AF+2 |
| **Group member details** | DV2016-0071 - Botha; Adam  C2SMCDZD3 - Ford; Ryan  69JJ1CZM8 - Kotze; James  DV2011-0590 – Morkel, Dreyer  DV2015-0655 - Prince; Jonathan  DV2015-0858 – van Zyl; Franco |
| **Project title** | A-Ford-Able Management System |
| **Submission date** | 02 November 2018 |
| **Signature of group leader** |  |

**Table of contents**

[1. Deliverable 1: Proposal 5](#_Toc528938299)

[1.1. Group and customer information 5](#_Toc528938300)

[1.2. Project introduction 6](#_Toc528938301)

[1.2.1. Background, purpose and scope 6](#_Toc528938302)

[1.2.2. Aim and objectives 6](#_Toc528938303)

[1.3. Requirements 7](#_Toc528938304)

[1.4. Schedule 8](#_Toc528938305)

[1.5. Risk management 13](#_Toc528938306)

[2. Deliverable 2: Methodology and development plan 17](#_Toc528938307)

[2.1. Methodology 17](#_Toc528938308)

[2.1.1. Iterative incremental development 17](#_Toc528938309)

[2.2. Development plan 19](#_Toc528938310)

[2.3. Schedule 20](#_Toc528938311)

[3. Deliverable 2: User Requirements Document (URD) 23](#_Toc528938312)

[3.1. Introduction 23](#_Toc528938313)

[3.1.1. Purpose 23](#_Toc528938314)

[3.1.2. Scope 23](#_Toc528938315)

[3.1.3. Definitions, acronyms and abbreviations 24](#_Toc528938316)

[3.1.4. Overview 24](#_Toc528938317)

[3.2. System requirements 25](#_Toc528938318)

[3.2.1. System perspective 25](#_Toc528938319)

[3.2.2. Functional requirements 25](#_Toc528938320)

[3.2.3. Non-functional requirements 30](#_Toc528938321)

[3.2.4. Technical requirements 32](#_Toc528938322)

[3.2.5. User characteristics 34](#_Toc528938323)

[3.2.6. Operational environment 34](#_Toc528938324)

[4. Deliverable 3: Information systems design 35](#_Toc528938325)

[4.1. Logical design 35](#_Toc528938326)

[4.1.1. Use case diagrams 35](#_Toc528938327)

[4.1.2. Class diagrams 46](#_Toc528938328)

[4.1.3. Activity diagrams 47](#_Toc528938329)

[4.1.4. Sequence diagrams 59](#_Toc528938330)

[4.2. Physical design 67](#_Toc528938331)

[4.2.1. Investigation of technologies to be applied 67](#_Toc528938332)

[4.2.2. System testing 68](#_Toc528938333)

[5. Deliverable 4: Implementation 72](#_Toc528938334)

[5.1. System interface design 72](#_Toc528938335)

[5.1.1. Iteration 1: Landing Page 72](#_Toc528938336)

[5.1.2. Iteration 2: New/Manage Job Page 74](#_Toc528938337)

[5.1.3. Iteration 3: Quote Page 75](#_Toc528938338)

[5.1.4. Iteration 4: Work Page 76](#_Toc528938339)

[5.1.5. Iteration 5: Finalise Page 77](#_Toc528938340)

[5.1.6. Iteration 1: Landing Page 78](#_Toc528938341)

[5.1.7. Iteration 2: New/Manage Job Page 84](#_Toc528938342)

[5.1.8. Iteration 3: Quote Page 86](#_Toc528938343)

[5.1.9. Iteration 4: Work Page 88](#_Toc528938344)

[5.1.10. Iteration 5: Finalise Job Page 90](#_Toc528938345)

[6. Deliverable 5: User manual 92](#_Toc528938346)

[6.1. Introduction 92](#_Toc528938347)

[6.2. How to install 92](#_Toc528938348)

[6.2.1. Double click on the Affordable MIS setup file 92](#_Toc528938349)

[6.2.2. Select Yes 92](#_Toc528938350)

[6.2.3. Select Create a desktop shortcut 93](#_Toc528938351)

[6.2.4. Select Next > 93](#_Toc528938352)

[6.2.5. Select Install 94](#_Toc528938353)

[6.2.6. Wait for setup to install application 94](#_Toc528938354)

[6.2.7. Select Finish 95](#_Toc528938355)

[6.2.8. Launch application 95](#_Toc528938356)

[6.3. Instructions 96](#_Toc528938357)

[6.3.1. Add a new client 96](#_Toc528938358)

[6.3.2. Search for a client 99](#_Toc528938359)

[6.3.3. Edit an existing client 102](#_Toc528938360)

[6.3.4. Manage an existing job 106](#_Toc528938361)

[6.3.5. View jobs per phase 108](#_Toc528938362)

[6.3.6. Add a new staff member 110](#_Toc528938363)

[6.3.7. Edit an existing staff member 113](#_Toc528938364)

[6.3.8. Add a new job 117](#_Toc528938365)

[6.3.9. Create a new quote 121](#_Toc528938366)

[6.3.10. Copy an existing quote 123](#_Toc528938367)

[6.3.11. Accept quote 125](#_Toc528938368)

[6.3.12. Reject quote 126](#_Toc528938369)

[6.3.13. Save quote to PDF 127](#_Toc528938370)

[6.3.14. Track work 128](#_Toc528938371)

[6.3.15. Track spend 129](#_Toc528938372)

[6.3.16. Add payments made 130](#_Toc528938373)

[6.3.17. Finalise job 132](#_Toc528938374)

[7. Deliverable 5: Evaluation report 133](#_Toc528938375)

[7.1. Introduction 133](#_Toc528938376)

[7.2. The final system and the customer’s requirements 134](#_Toc528938377)

[7.3. The methodology 135](#_Toc528938378)

[7.4. The customer 136](#_Toc528938379)

[7.5. Group dynamics and team collaboration 139](#_Toc528938380)

[7.6. Time management 142](#_Toc528938381)

[7.7. Lessons learned 145](#_Toc528938382)

[7.8. Conclusion 146](#_Toc528938383)

[8. Customer sign-off 147](#_Toc528938384)

[9. Bibliography 148](#_Toc528938385)

1. Deliverable 1: Proposal
   1. Group and customer information

Group: 1

Group Name: Awesome Foursome Plus 2 (AF+2)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Surname | Student Number | Skills |
| Adam | Botha | DV2016-0071 | Software Development |
| Ryan | Ford | C2SMCDZD3 | Project Management |
| Dreyer | Morkel | DV2011-0590 | Document Management |
| James | Kotze | 69JJ1CZM8 | Interface Design |
| Franco | van Zyl | DV2015-0858 | Logical Design |
| Jonathan | Prince | DV2015-0655 | Physical Design |

**Customer Details:**

|  |  |
| --- | --- |
| Owner Name | Ricky Ford |
| Company Name | Sevlac Investments |
| Industry | Investment |

* 1. Project introduction
     1. Background, purpose and scope

**Background**

The company we have chosen to develop a system for is Sevlac investments. They have invested into a small business that specialises in general handyman work such as painting, plumbing, tiling, roofing, security gates, alarm systems, JoJo tank implementation and repairs, irrigation systems, electrical appliances repairs and implementation. The company's name is A-Ford-Able Handyman Services and was established on the 21st of January 2001. A-Ford-Able Handyman Services has been in high demand since its opening day and the CEO Mr. Ricky Ford has decided that it is time to improve the business. Mr Ford has chosen to improve his business through the implementation of an ERP management system for himself and his employee Mr Steven Ford who is the manager of the business.

**Purpose**

Mr Ricky Ford has specified the main need for the system is to help both himself and Mr Steven Ford to improve workflow, profits and manage deadlines better. Both Mr Ricky and Mr Steven Ford have different needs for their system and their needs and wants for their interfaces are both different. For example, Mr Ricky Ford has requested that the interface should be more focused around the triple constraint theory therefore scope, schedule and cost. Mr Steven Ford works on site with the employees and therefore needs a system to help him see upcoming due dates, inventory, available employees, and pricing for materials. Steven Ford should also be able to use the system to generate quotes for customers, as well as manage these quotes.

**Scope**

The scope of the project is going to be broken down into six deliverables each with different goals and objectives each approved and signed off by the client Mr Ricky Ford. The project’s start date is **02/01/2018** and the due date is defined as **20/09/2018.** Deliverable one’s due date is **19/03/2018**. Deliverable two’s due date is **07/05/2018.** Deliverable three’s due date is **06/08/2018.** Deliverable four’s due date is **03/09/2018.** Deliverable five’s due date is **08/10/2018.** Deliverable six’s due date is **29/10/2018.**

* + 1. Aim and objectives

**Single aim of the project**

The single aim of the project is to meet all of the clients requirements along with the predefined due dates.

**Objectives of the project**

* Have all the requirements stated at the start of the projects
* Have all requirements understood at the start of the project
* Have the requirements rated on a scale that is both needed and possible to achieve for the project
* Have all deliverables signed off by the clients
* Have all deliverables completed on the predefined dates
  1. Requirements

Functional requirements:

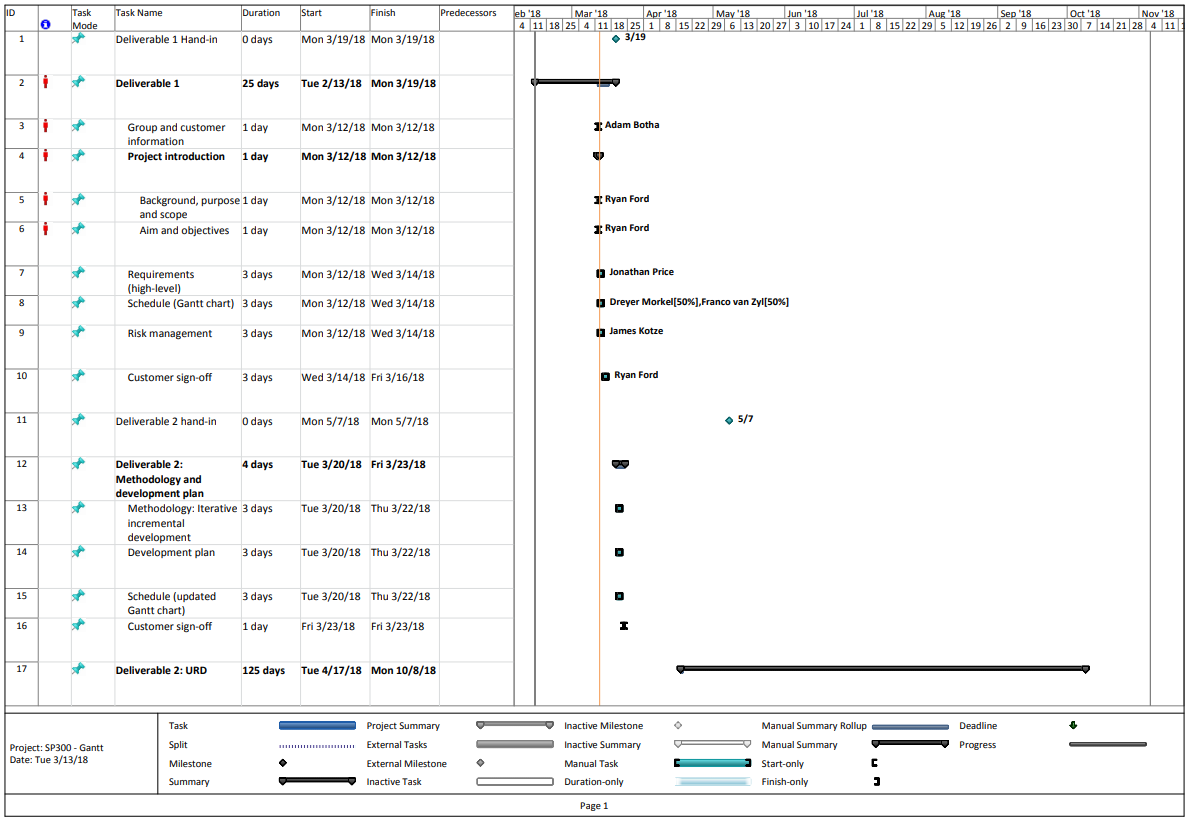
* Enable the user to login before using the application
* List all active jobs at the current time
* Change the job specification due to client request or due to employee’s mistake
* Flag irregularities due to changes been made.
* Checkout of various equipment
* Forecast income and expenses based on job input
* Produce various report diagram to report(Graphs)
* Create a new invoice
* Update and add to existing invoice
* Issue a calculate quote for the client
* Store quotes from suppliers which is used to calculate quote for client
* Compile a site report to be used for quotation
* Track employees hours worked and salary due
* Save signed copies of each client’s indemnity form
* Budget the hours worked for each job
* Save all the relevant information for each employee
* Store each client’s information to be reference to
* Notify user about time or resource conflicts
* Create schedule for each job
* Manage and assign resources
* Post job site inspection to ensure job is done

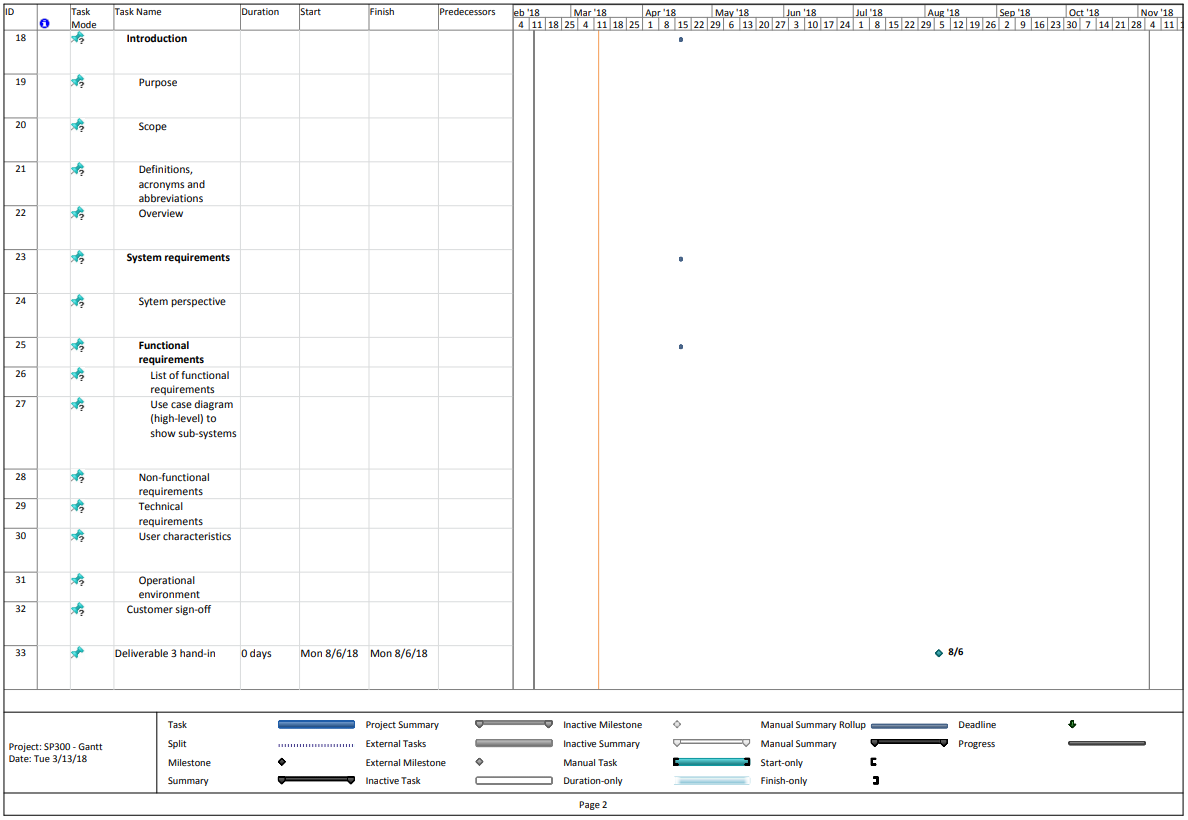
Non-Functional requirements:

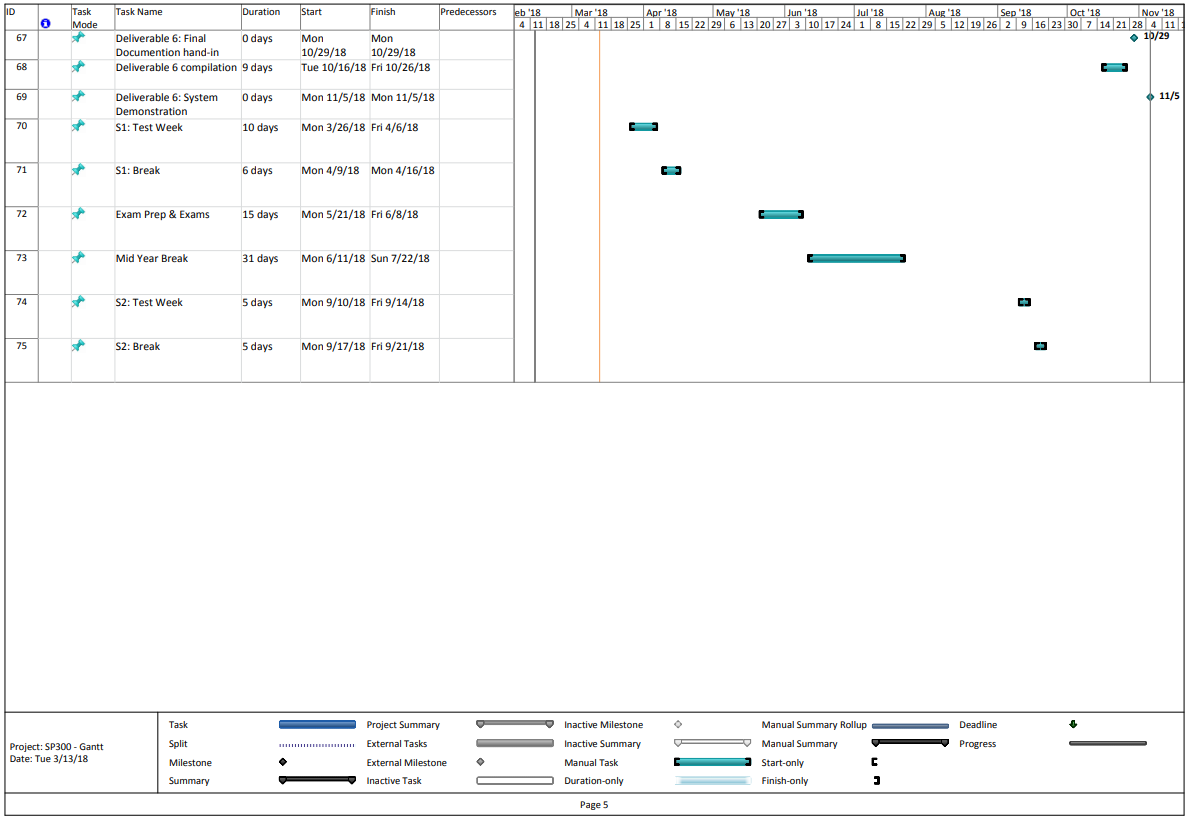
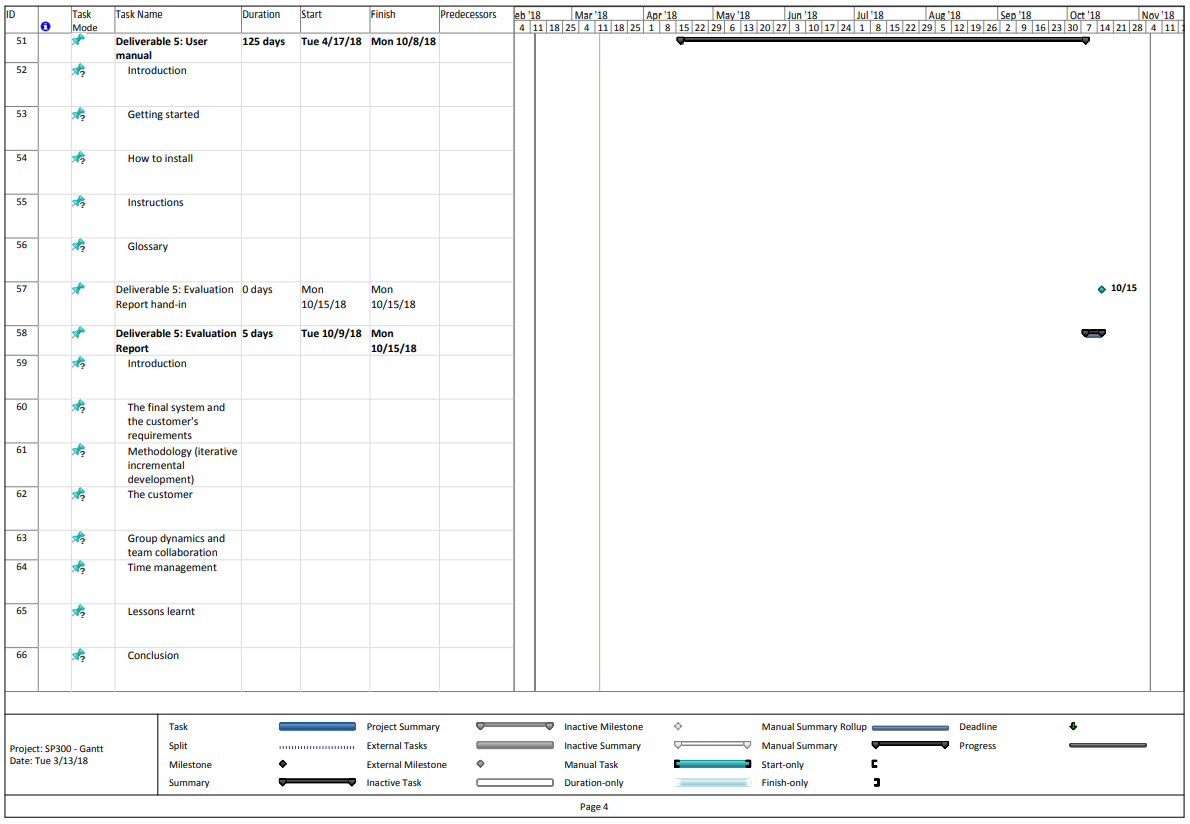
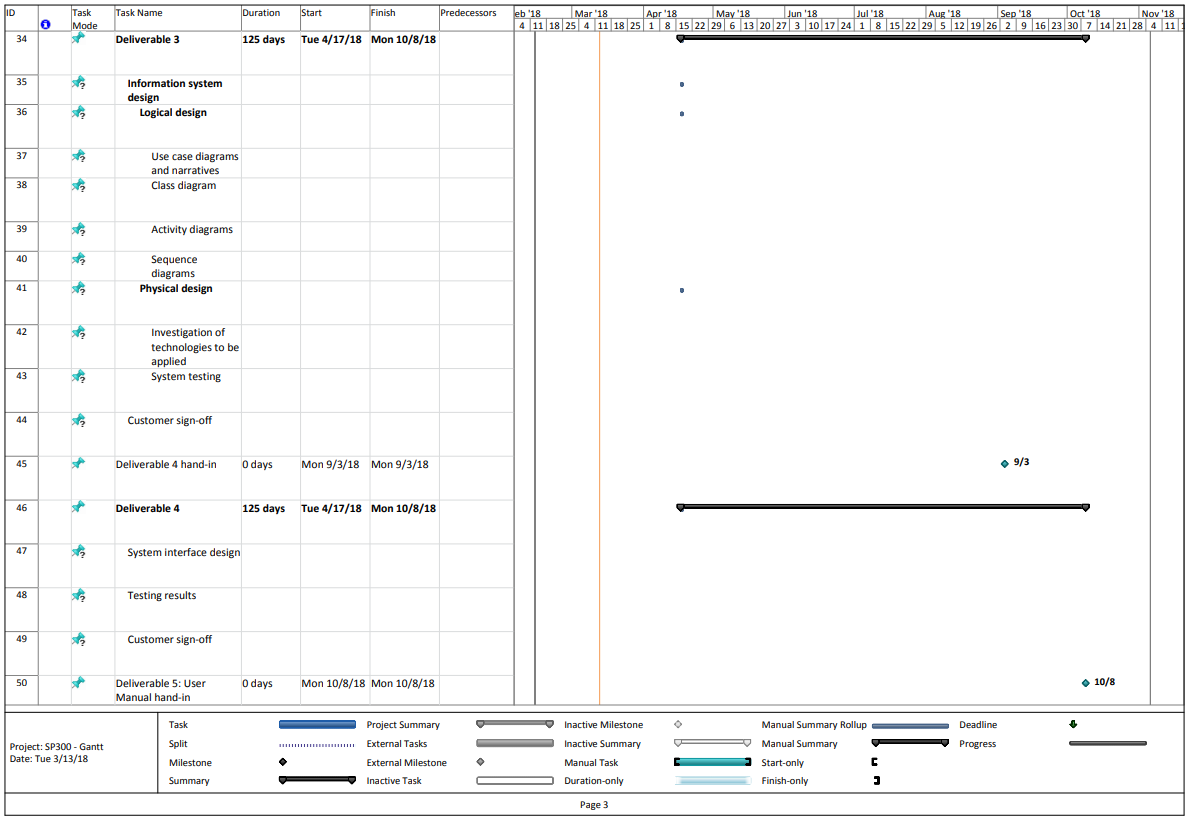
* System works with user accounts
* Log activities/changes as they are made
* Changes must be cascaded through the whole system.
* Keep records of all invoices
* Track job progress
* Provide special forms for various scenarios.

Technical requirements:

* The software should be able to run on a computer with:
  + Microsoft Windows 7 or 10
  + 2GB RAM
  + Dual Core processor
  + 100MB free hard drive space
  + Latest version of Java
  1. Schedule







* 1. Risk management

Risk management focuses on identifying and responding to risks that might or might not occur through the project lifecycle.

Evaluating risks and managing them are key to project planning and project success. Risk management techniques helps the project manager to evaluate the team’s response to risks, as well as evaluate the current responses and practices put in place.

There are several response strategies towards risks:

* **Mitigate**: Reduce the impact that the risk might have
* **Avoid**: Try to avoid the risk from occurring at all cost (usual responses to high impact high probability risks)
* **Transfer**: Try to transfer the effects of the risk towards another party or area of the organisation to reduce the impact
* **Accept**: Accept the fact that the risk will occur and the impact thereof (which would usually be the response to low impact and high probability risks)

**Risk register**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Risk | Description | Category | Root Cause/Triggers | Potential Response | Risk Owner | Probability | Impact | Status |
| R01 | Poor Scope | Scope is not defined properly and not well received from client. | People | Poor Communication or understanding between client and group | Mitigate | Ryan Ford (PM) | Low | High |  |
| R02 | Time estimation | Poor estimation and scheduling from the group. Could possibly be incomplete or late deliverables – do to not following set schedule. | People | Group not completing work in acquired time.  Poor estimation of time and poor scheduling.  Client changes Time constraints. | Mitigate | Dreyer Morkel | Low | High |  |
| R03 | Poor Quality | Final deliverables are of poor quality and completed work is not up to expected standards | People | Inadequate team members. | Avoid | Ryan Ford (PM) | Low | High |  |
| R04 | Lack of communication | If there is a lack in communication between the client and the team. Or a lack in communication between the team members | People | Poor communication techniques used amongst team members | Mitigate | Ryan Ford (PM) | Low-Med | Med-High |  |
| R05 | Poorly addressed risks | When risks are ignored or not addressed properly as attention requires. | People | Team not realizing the importance of risk management | Avoid | Ryan Ford (PM) | Med | Med-High |  |
| R06 | Hardware failure | If the technology we work on fails (e.g. technical error) and data is lost accordingly (e.g. old or faulty hard drive fails) | Technology | Old / incompatible / faulty hardware or software | Avoid | Franco van Zyl | Low | High |  |
| R07 | Poor data management | When data security policies (e.g. regular backups) are not properly implemented, loss of data can occur in congruence with risk (ID = R06) when lost data cannot be recovered. | Structure  Process  People | Poor data management and security | Avoid | Franco van Zyl | Low | High |  |
| R08 | Incomplete work | If one of the team members responsible for completing work regarding a deliverable is injured or sick to the point where he/she might not be able to complete the work in time. | People | Team member injury or illness | Transfer | Adam Botha | Low | Med-High |  |

**Risk Probability and Impact**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Low | Medium | High |  |
| Probability |  |  |  | High |
|  | R04  R05 |  | Medium |
|  | R08 | R01  R02  R03  R06  R07 | Low |
|  | Impact | | |  |

1. Deliverable 2: Methodology and development plan
   1. Methodology
      1. Iterative incremental development

The process starts with planning and will go through iterative developmental cycles. Technopedia (2018) states that during each cycle, additional features are added and existing ones are upgraded, resulting in a completed piece of software being released after each iteration. The end user is continuously involved and their feedback used to alter the list of requirements, features and fixes the next version of the software will include.

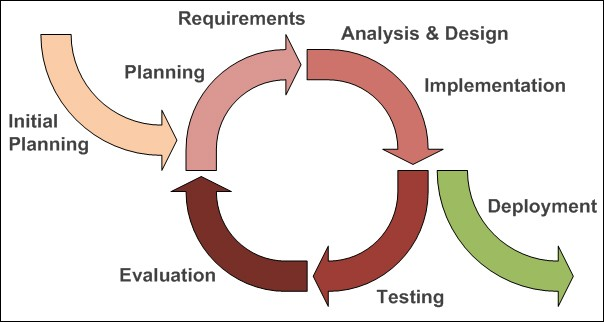
Ghahrai (2017) notes that this development strategy consists of two parts: incremental and iterative. The first suggests that the system components are constructed independently and added to a little bit at a time. The components are included into the final product when they are completed. The second suggests that the system as a whole is continuously evaluated and improved until it is finished. The two processes combined results in a methodology where the system is broken into components. The components can be worked on separately and each improved upon. The components, after any successive step has been completed, can be combined to form a complete and functioning system which fulfils some, and eventually all, of the user requirements. Developers can use and test the program after each cycle to determine where to improve, fix and add features to, the software. Only that which is needed to successfully deliver the next iteration is considered and invested in (Farcic, 2014).

This methodology is superior to linear workflows, like the waterfall model. Farcic (2014) gives a few reasons as to why this methodology is better:

* it allows for errors to be detected early
* changes can be made more easily
* it is easier to satisfy the end user’s needs because they can give feedback more times during development
* the cycle is executed faster each time as team members become accustomed to the repeating workflow and development time is shortened
* team knowledge as a whole is increased because the members work closely together
* requirements can change as the situation changes

Technopedia (2018) proposes that the methodology has the following phases:

1. **Inception**: Determine the project scope, the requirements that need to be realised and the risks involved.
2. **Elaboration**: Decide on and create a software architecture that reduces the risks mentioned and also satisfies non-functional requirements.
3. **Construction**: The architecture components are built up in increments, producing usable, error-free code each time. The increments cycle through the traditional steps of analysis, design, development, implementation and testing of the functional requirements.
4. **Transition**: The final product is installed, run and used by the end user.



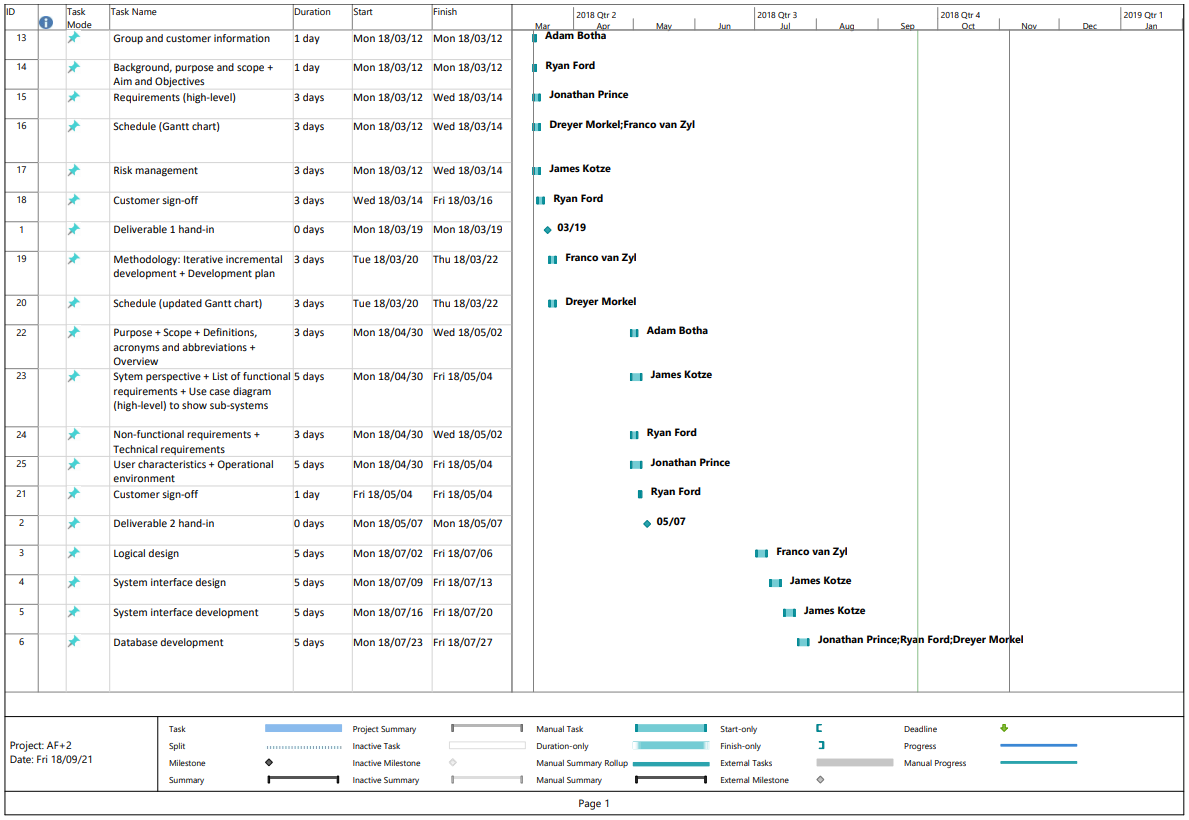
<https://www.researchgate.net/profile/A_B_M_Moniruzzaman/publication/249011841/figure/fig2/AS:341184099700745@1458356036923/Iterative-and-incremental-agile-development-process-source.jpg>

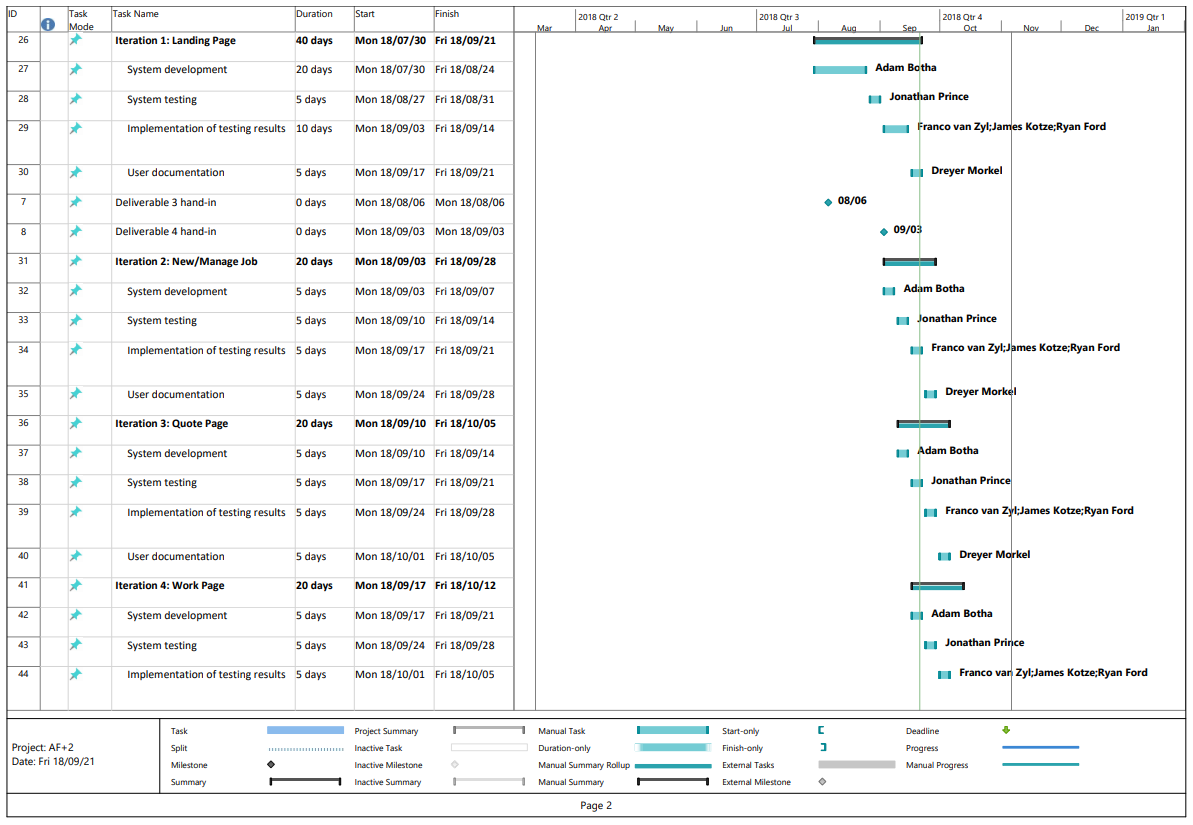
An example from real life is the evolution of the mobile phone. The first ever phone had all the parts to do what it needed to do, even though those parts are, by current standards, outdated. You could also only voice call someone. Then people thought about how to make the components faster, smaller and more powerful and make a mobile phone do more (SMS). After each new version there existed a working device with superior parts and functionality. Look at what we have today - and we are still improving!

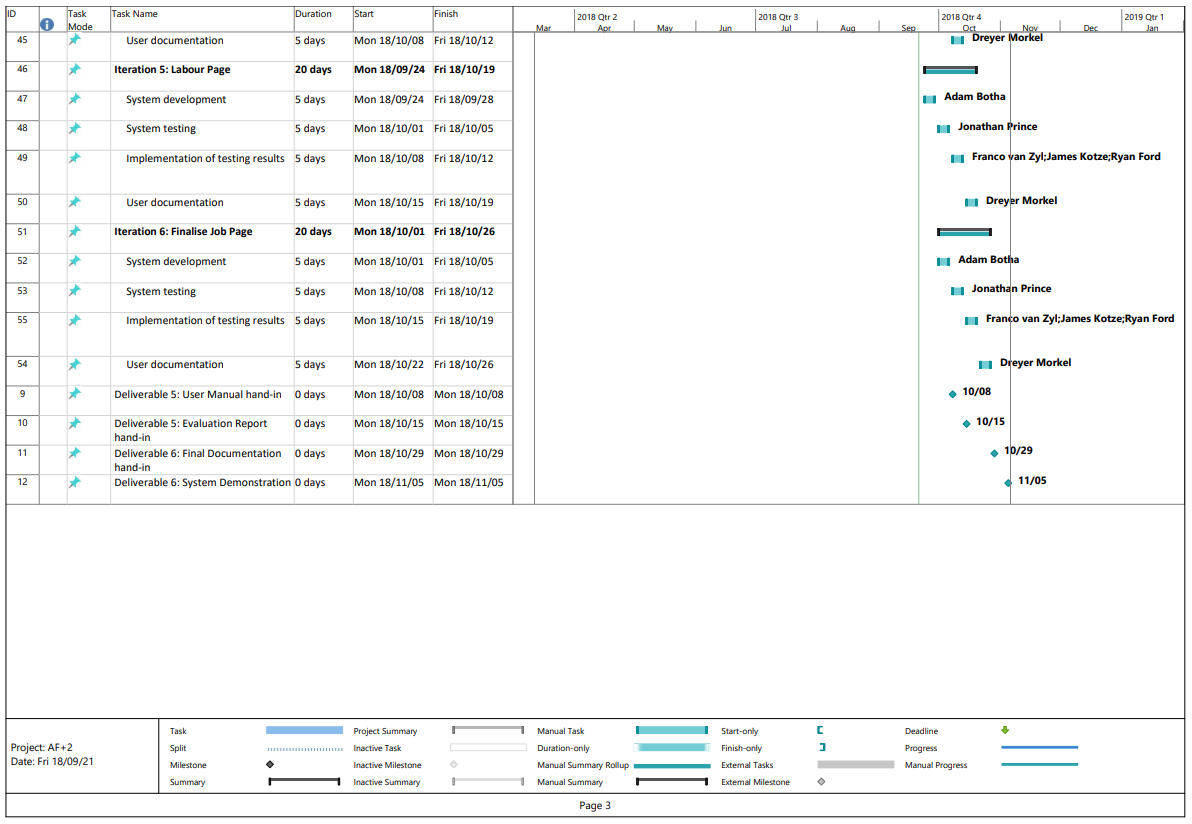
* 1. Development plan

|  |  |  |
| --- | --- | --- |
| Increment | Functional requirement | Time-frame |
| 1. Landing Page | * Add a new client * Edit an existing client * View clients * View jobs per client * Search for clients * View jobs per phase * Add a new staff member * Edit an existing staff member * View staff members | 8 weeks |
| 2. New/Manage Job Page | * Add a new job * Manage an existing job * Update job details | 4 weeks |
| 3. Quote Page | * Create a new quote * Create a copy of a quote * Reject a quote * Accept a quote * Print a quote | 4 weeks |
| 4. Work Page | * Add materials * Add overheads * Add labour * Track spend | 4 weeks |
| 5. Finalise Job Page | * Track spend * Add payments made * Finalise job | 4 weeks |

* 1. Schedule







1. Deliverable 2: User Requirements Document (URD)
   1. Introduction
      1. Purpose

The purpose of this document is to portray our understanding as a team of what the user expects from the final system. This basically means that we will outline how we understood the requirements by listing them in three different categories namely: Functional, Non-functional and Technical requirements. This will outline everything the user expects the system to be able to do. This document however will be open to the client’s comments and will change if we (together with the client) identify a requirement that was missed or badly understood. We will also outline other aspects of the system including user characteristics and user environment (which is basically the environment that the system will be used in on a day to day basis). The end goal of this project is to have a system that the client has requested and that meets all the requirements and specifications that have been stated and produced within the predetermined time frame.

* + 1. Scope

**System Name:**

A-Ford-Able Management System:

**What the system will do:**

The system will be used by our client to manage and keep track of his businesses. This will include functions like keeping track/recording of financial statements that indicate the total money spent and earned. The system will also be used to keep track of the inventory of the tools related to the business in order to make sure that tools are only bought when they are not yet owned. Furthermore the system will also have a component dedicated to scheduling appointments with clients and setting deadlines for project completion.

**Benefits and goals related to the system:**

The system we will be developing will have many advantages. Firstly, by keeping track of financial records, it will allow our client to better manage the expenses of the business. Even more money will possibly be saved on equipment costs. Our client has noted to us that the employees of the business often do not know what equipment they do and do not have, which results in them buying new equipment when they need it. By implementing our system, the employees of the business can build up an inventory of tools/equipment they already own, which can be used to plan for upcoming jobs they need to complete. It will improve the productivity of the business.  Furthermore, as stated above, the system will allow the employees of the business to schedule appointments. This will allow for a more effective use of their time, as they will know in advance what job needs to be done and where. This in turn will allow them to fit in more clients than usual and as a result the company's revenue will likely increase.

* + 1. Definitions, acronyms and abbreviations

Currently no acronyms or abbreviations have been used and the language used does not require any definitions to be more extensively explained as we have used a simplistic and effective means to complete these deliverables so that all that read it have the ability to understand it.

* + 1. Overview

In section two this document will go over the various requirements that were captured in our interviews with the client. The requirements have been divided into 3 main subsections. These include functional-, non-functional- and technical requirements. The next section will also include/explain the system perspective, user characteristics and the operational environment. All of which will help both us as developers and the client to better understand the system and the systems feasibility and requirements.

* 1. System requirements
     1. System perspective

The system is mostly stand-alone as it will only interact with a database, which during development will be developed using technologies such as MySQL. The database will be hosted locally so connection and interaction with this database will not require any specific 3rd party system. We aim to develop this system as clean-cut and stand-alone as possible to ensure a custom and specific product that does not require extra technologies unless it is absolutely necessary.

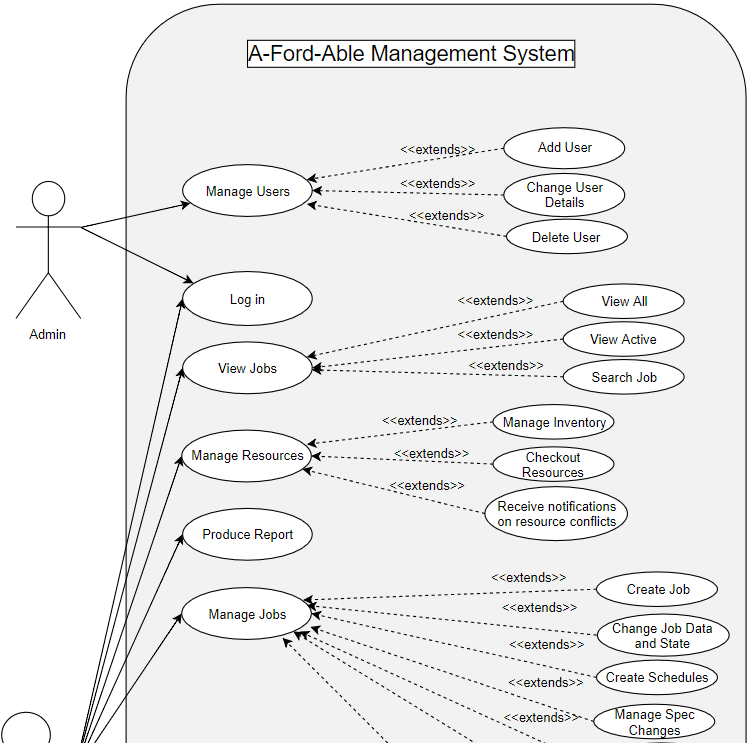
* + 1. Functional requirements
       1. List of functional requirements
* Enable the user to login before using the application
  + Store User data
* View Jobs
  + List all jobs
  + List all active jobs at the current time (or specific time)
  + Search jobs by identifier
* Produce various report diagram to report (Graphs)
* Manage Jobs
  + Create Job
  + Change Job State and data (active, employees, completed, etc.)
  + Create schedules for job
  + Manage the job specification changes
  + Flag irregularities due to changes been made.
  + Post job site inspection to ensure job is done
  + Manage employees on job
* Manage Employees
  + Store employee data (create, edit, remove)
  + Track employees hours worked and salary due
* Manage Resources
  + Manage and assign resources
    - Store resource inventory
  + Checkout of various equipment
  + Notify user about time or resource conflicts
* Manage Finances
  + Outgoing Finances
    - calculate expenses for company (per job and collectively per month)
      * Human resources
      * Physical resources
  + Store quotes from suppliers
  + Quotations
    - Budget estimated work hours for job
    - Compile a site report (also used for quotation)
    - Issue calculated quote for the client
  + Invoices
    - Budget Actual work hours for job
    - Create a new invoice
    - Change existing invoice
  + Forecast income and expenses based on job input
* Manage Clients
  + Store client data
  + Change client data
  + Store signed copies of each client’s indemnity form

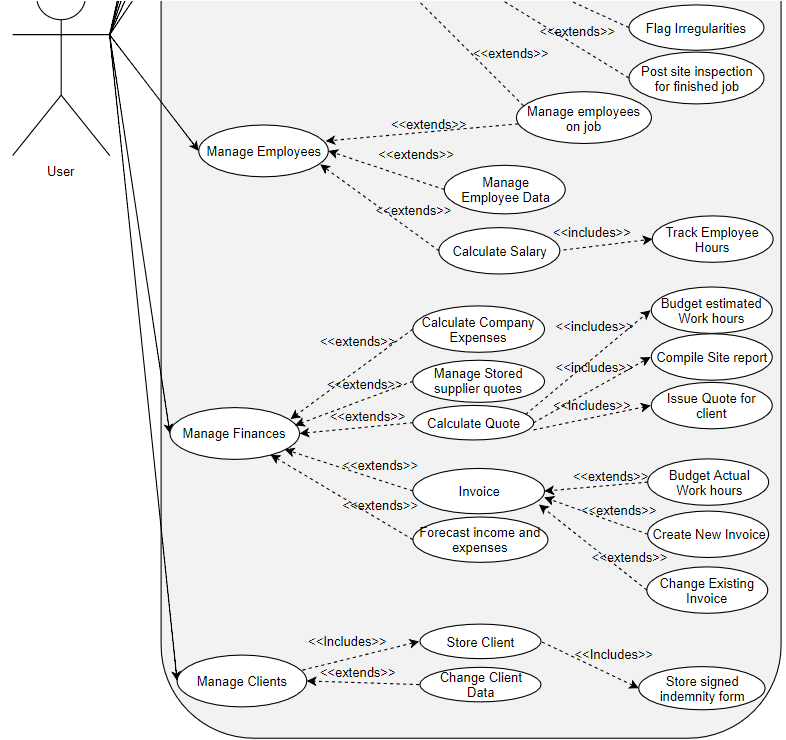
**MoSCow rating system for functional requirements:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Requirement | Must have | Should have | Could have | Would have |
| A1 | Store User data | X |  |  |  |
| A2 | List all jobs | X |  |  |  |
| A3 | List all active jobs at the current time (or specific time) | X |  |  |  |
| A4 | Search jobs by identifier |  |  | X |  |
| A5 | Produce various report diagram to report (Graphs) |  | X |  |  |
| A6 | Create Job | X |  |  |  |
| A7 | Change Job State and data (active, employees, completed, etc.) |  | X |  |  |
| A8 | Create schedules for job |  | X |  |  |
| A9 | Manage the job specification changes |  | X |  |  |
| A10 | Flag irregularities due to changes been made. |  | X |  |  |
| A11 | Post job site inspection to ensure job is done | X |  |  |  |
| A12 | Manage employees on job |  | X |  |  |
| A13 | Store employee data (create, edit, remove) | X |  |  |  |
| A14 | Track employees hours worked and salary due | X |  |  |  |
| A15 | Manage and assign resources |  | X |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Requirement | Must have | Should have | Could have | Would have |
| A16 | Checkout of various equipment |  | X |  |  |
| A17 | Notify user about time or resource conflicts |  | X |  |  |
| A18 | Store quotes from suppliers |  |  | X |  |
| A19 | Budget estimated work hours for job |  |  | X |  |
| A20 | Compile a site report (also used for quotation) |  | X |  |  |
| A21 | Issue calculated quote for the client |  | X |  |  |
| A22 | Forecast income and expenses based on job input |  | X |  |  |
| A23 | Store client data | X |  |  |  |
| A24 | Change client data | X |  |  |  |
| A25 | Store signed copies of each client’s indemnity form | X |  |  |  |

* + - 1. Use case diagram to show sub-systems





* + 1. Non-functional requirements

Provide a list of the non-functional requirements. The following categories could be used:

* Human-computer interactions (user interfaces)
* System works with user accounts
* User accounts can be updated and customized at the user's discretion
* Simplistic layout
* Intuitive design
* Lots of user feedback on actions
* Provide special forms for various scenarios
* Changes must be cascaded through the whole system
* Security requirements
* Log activities/changes as they are made
* Keep records of all invoices
* Maintain integrity of all invoices
* Maintain integrity of all data stored
* Changes must be authorised and logged
* Communications interfaces
* Log activities/changes as they are made
* Notifications of job alerts
* Hardware and software interfaces
* Computer mouse
* Keyboard
* Monitor
* Printer
* Other performance requirements: speed, maintainability,, robustness, flexibility, etc.
* Accessibility
* Well-designed external API with support of authentication and authorization
* High performance API
* Extensive developer documentation
* Scalability
* Able to handle high traffic
* Able to integrate and store data on a database
* Able to handle peak spike connections
* Reliability
* System should have 99% uptime
* System should have minimal downtime

**MoSCow rating system for non-functional requirements:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Requirement | Must have | Should have | Could have | Won't have |
| B1 | System works with user accounts | X |  |  |  |
| B2 | User accounts can be updated and customized at the user's discretion | X |  |  |  |
| B3 | Simplistic layout |  | X |  |  |
| B4 | Intuitive design |  | X |  |  |
| B5 | Lots of user feedback on actions |  |  | X |  |
| B6 | Provide special forms for various scenarios |  | X |  |  |
| B7 | Changes must be cascaded through the whole system |  | X |  |  |
| B8 | Log activities/changes as they are made |  | X |  |  |
| B9 | Notifications of job alerts |  | X |  |  |
| B10 | Computer | X |  |  |  |
| B11 | Mouse | X |  |  |  |
| B12 | Keyboard | X |  |  |  |
| B13 | Printer | X |  |  |  |
| B14 | Well-designed external API with support of authentication and authorization |  | X |  |  |
| B15 | High performance API | X |  |  |  |
| B16 | Extensive developer documentation | X |  |  |  |
| B17 | Able to handle peak spike connections | X |  |  |  |
| B18 | System should have 99% uptime | X |  |  |  |
| B19 | System should have minimal downtime | X |  |  |  |

* + 1. Technical requirements

The software should be able to run on a Microsoft Windows 7/10 machine with 2GB RAM and a Dual Core processor. These requirements are fairly low to ensure that the system can run almost anywhere, even if the client only has a low spec PC available.

* Operating system
* Microsoft Windows 7
* Microsoft Windows 10
* Hardware
* 2GB ram
* Dual core processor
* Software
* Java
* Microsoft Word
* Microsoft Project
* Google Drive
* Google Docs
* Programming language
* Java
* SQL
* Database application
* Microsoft Access
* Architecture
* Integrated system architecture
* Model-View-Controller (MVC)

**MoSCow rating system for technical requirements:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Requirement | Must have | Should have | Could have | Won't have |
| C1 | Windows 7 |  | X |  |  |
| C2 | Windows 10 |  | X |  |  |
| C3 | 2GB ram | X |  |  |  |
| C4 | Dual core processor | X |  |  |  |
| C5 | Java | X |  |  |  |
| C6 | Microsoft Word | X |  |  |  |
| C7 | Microsoft Project | X |  |  |  |
| C8 | Google Drive |  |  | X |  |
| C9 | Google Docs |  |  | X |  |
| C10 | Java | X |  |  |  |
| C11 | SQL | X |  |  |  |
| C12 | Workbench |  | X |  |  |
| C13 | PhpMyAdmin |  | X |  |  |
| C14 | Xampp |  | X |  |  |
| C15 | Integrated system architecture | X |  |  |  |
| C16 | Model-Controller-View (MVC) |  | X |  |  |

* + 1. User characteristics

The system that is being developed has only two users who will initially access the system however the system should be developed with the possibility of growing the system and therefore have more user. The two users will be the boss and the foreman, however as mentioned a third type of user will need to be considered this will be the general worker.

The boss have a decent understanding of the system and is considered as an expert as the boss has many years of experience on working on various systems of the same type. The Boss will access the system more or less one a month with the sole purpose of checking up on the foreman as the boss is a busy man and does not have much time to spend on the system. Thus the boss will be interested in various progress reports giving a summary what the foreman has completed, what the foreman is busy working on and what work is there for the future.

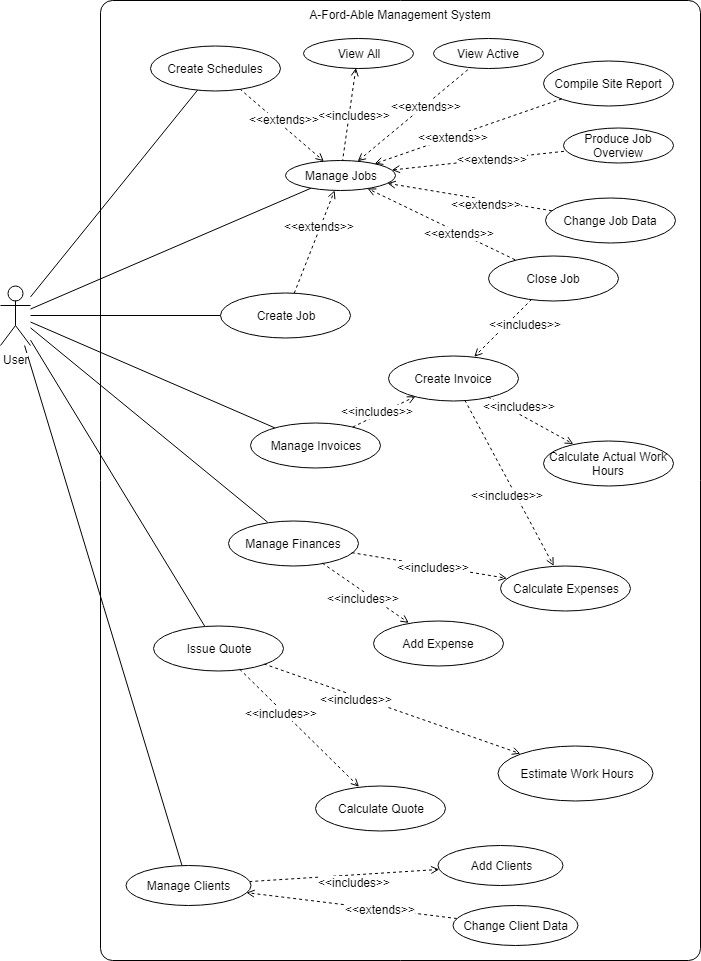
The foreman is considered to be not as computer literate and has little to no experience working on a computer. However the foreman will be the person who will interact with the system the most. The foreman will be logging quotations, jobs and expenses. Therefore the foreman will spend a lot of time on the system. The foreman will also access the system from various location like the construction site or the office.

The workers might, in the future, use the system to access the work set out for them and to report various things that may occur.

* + 1. Operational environment

The system will have to run on various type of devices as the system may have to be accessed from multiple locations at once. The system will mainly be run on a desktop pc for example the pc in the office which will always be stationary and is protected and secured, therefore it will not be as big of a risk. The system will also run on a mobile device as the system will need to be accessed from the construction site. As this pose various security threats the system needs to be secured as the device maybe lost and the system may be exposed. The boss may also need to access the system from abroad using a laptop or any portable device and therefore the system should not be limited to local areas.

1. Deliverable 3: Information systems design
   1. Logical design
      1. Use case diagrams



**Use Case Name: Manage Jobs**

Brief: The user opens the program and chooses to manage jobs. From there the user can perform all general (add, remove, edit, view) and job-specific activities like creating invoices and quotes, adding expenses and managing a calendar.

Assumptions: The user has started the program and is on the main screen.

Normal Flows (N)

1. The user chooses to 'Manage Jobs'.
2. The software opens a new window with a list of all the jobs in the database.
3. The user selects a job from the list and chooses to view the job.
4. The software opens a new window displaying the details of the job. The user has similar options to those in the 'Job Manager' window.
5. The user closes the window.
6. The software returns to the 'Job Manager' window.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: Create Job**

Assumption: The user has the 'Job Manager' window open.

Brief: The user enters data for a new job and the system adds the job to the database.

Normal Flows (N)

1. The user chooses to add/create a new job.
2. The software opens a new window with data entry fields.
3. The user enters data for the job. (E1)
4. The software validates the entered data and it is correct.
5. The user adds/selects a client for the job (S1).
6. The user sets a start and end date for the job. (S2)
7. The user accepts the new job.
8. The software adds the new job to the database.
9. The data entry window closes.
10. The new job is added to the job list in the 'Job Manager' window.

Subordinate Flows (S)

S1

1. The software prompts the user to select an existing client (E2) or create a new one.
2. The software opens a new 'Client' data entry window.
3. See Create Client, go to N4.

S2

1. See Create Schedules, go to S1.3

Alternate/Exceptional Flows (E)

E1

1. The user enters incorrectly formatted or typed data.
2. The software indicates this to the user.
3. The user makes corrections.
4. The software validates the entered data again.

E2

1. The software presents the user with a list of all existing clients.
2. The user selects a client from the list and accepts it.
3. The client is linked to the job.

**Use Case Name: Close Job**

Assumption: The user has the 'Job Manager' view open.

Brief: The job has been completed and the owner has to be issued an invoice and the job status changed to 'pending payment'.

Normal Flows (N)

1. The user selects the job from the list they want to close.
2. The user chooses to close the job.
3. The software changes the job status to 'PENDING PAYMENT'.
4. The software sets the end date of the job to the closing date.
5. The software opens a new window displaying the invoice data for the job.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: Change Job Data**

Assumption: The user has the 'Manage Job's view open and has selected the job they want to change the data of.

Brief: The user selects a job to possibly alter all of the data set when it was created.

Normal Flows (N)

1. The user chooses to edit the job.
2. The software opens a new window with data entry fields containing the current data for the selected job.
3. The user makes valid changes to the data (E1).
4. The software validates the changes.
5. The user accepts the changes.
6. The software updates the database.
7. The data entry window is closed.
8. The job list on the 'Job Manager' window is updated with the altered job.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

E1

1. The user enters incorrectly formatted or typed data.
2. The software indicates this to the user.
3. The user makes corrections.
4. The software validates the entered data again.

**Use Case Name: Produce Job Overview**

Assumption: The user is on the 'Job Manager' window and has a job selected from the list.

Brief: The user wants to see a complete overview of a job, including all its related entities.

Normal Flows (N)

1. The user chooses to produce a job overview.
2. The software opens a new window, gathers all entities related to the job, formats the connection and informational data and presents it to the user.
3. The user closes the job overview.
4. The software returns to the 'Job Manager' window.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: View Active**

Assumption: The user has the 'Job Manager' window open.

Brief: The user wishes to see al jobs that have their status set to 'ACTIVE'.

Normal Flows (N)

1. The user adjusts the filter properties to select active jobs.
2. The software searches for and displays only the jobs that are active in the job list (E1).

Subordinate Flows (S)

S1 - The software notifies the user that there are no jobs in the database.

S2 - The software notifies the user that there are no jobs with the status of 'ACTIVE'.

Alternate/Exceptional Flows (E)

E1

1. There are no jobs in the database (S1) or no jobs that are active (S2).

**Use Case Name: View All**

Brief: The default filter setting when opening the 'Job Manager' window.

Normal Flows (N)

1. The user opens the 'Job Manager' window. (E1)
2. The software displays the window with a list of all jobs in the database.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

E1

1. The user changes the filter settings to display all jobs.
2. The software adds all the jobs in the database to the job list.

**Use Case Name: Create Schedules**

Brief: The user wants to create a schedule for a job and entities belonging to the job.

Normal Flows (N)

1. The user has the main screen open and either goes to the 'Job Manager' (S1) or the 'Calendar Manager' (S2).

Subordinate Flows (S)

S1

1. The software opens the 'Job Manager' window.
2. The user selects a job from the job list and chooses to manage its schedule.
3. The software opens up a 'Calendar Item' data entry window with the job connected by default (E1).
4. The user sets data for the calendar item.
5. The software validates it.
6. The user accepts it.
7. The software saves the new calendar item to the database.

S2

1. The software opens the 'Calendar Manager' window.
2. All jobs that have start and end dates/times are shown to the user (in list form for now, descending in order of start date).
3. The user chooses to create a new calendar item (schedule).
4. The software opens a 'Calendar Item' data entry window.
5. The user enters data for the calendar item, also specifying the job (S3) and human resource (E1).
6. The software validates the data.
7. The user accepts it.
8. The software saves the new calendar item to the database.
9. The window closes and the list of calendar items in the 'Calendar View' is updated.

S3 - sets a start and end date for the job itself only

1. The software presents a list of all non-scheduled jobs.
2. The user selects a job from the list.
3. The software links the job to the calendar item.

Alternate/Exceptional Flows (E)

E1 - With both a job and a human resource, the latter has a start and end date for the job.

1. The software presents a list of all the human resources.
2. The user selects a human resource to schedule dates/times on the job for.
3. The software links the human resource to the calendar item for the job.

**Use Case Name: Manage Invoices**

Assumption: The user has the 'Job Manager' window open.

Brief: The user wants to view the invoice of any job. An invoice is produced from current job data and is therefore except from change by the user directly.

Normal Flows (N)

1. The user selects a job from the job list and chooses to view its invoice. (E1)
2. The software opens a new window, gathers all related job information, makes calculations and produces a formatted invoice of the job as it currently is.
3. The user closes the invoice window.
4. The software returns to the 'Job Manager' window.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

1. The software determines that the job's state is not 'PENDING\_PAYMENT' and informs the user that an invoice cannot the created.

**Use Case Name: Calculate Actual Work Hours**

Brief: The user has selected to create an invoice for a job and the software has to calculate human resource costs for the finished job.

Normal Flows (N)

1. The software inspects the job's schedule and selects all the human resources that worked on it.
2. The software calculates the total time each human resource worked and adds this to a total.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: Create Invoice - See 'Manage Invoices'**

**Use Case Name: Issue Quote**

Assumption: The user has created a job with the status 'INSPECTION' or 'ON\_HOLD\_PENDING\_APPROVAL' and now wants to quote the job.

Brief: The user creates a quote for the job, provides general data and adds several items to the quote with more specific data.

Normal Flows (N)

1. The user selects a job from the job list in the 'Job Manager' window. (E1)(E2)
2. The user chooses to create a quote for the job.
3. The software opens a new data entry (Quote) window for the quote.
4. The user enters quote data. (S1)
5. The software validates the data entered.
6. The user accepts the quote.
7. The software stores the quote in the database. If started in 'Quote Manager' window, E3.

Subordinate Flows (S)

1. The user chooses to add an item to the quote.
2. A data entry window appears.
3. The user enters data for the quote item.
4. The user accepts the quote item.
5. The software adds the item to the database.
6. The newly created quote item is added to the quote item list of the parent quote.
7. The window closes and the user is returned to the Quote window.

Alternate/Exceptional Flows (E)

E1 - The user is already viewing a job. (N2)

E2

1. The user has the 'Quote Manager' window open.
2. The user chooses to create a new quote. (N3)

E3 - The software adds the new quote to the quotes list of the 'Quote Manager' window.

**Use Case Name: Calculate Quote**

Brief: When the user wants to view a quote, all the quote items need to be retrieved and totals calculated.

Normal Flows (N)

1. The user is either in the 'Quote Manager' window or in the 'Job Manager' window (E1).
2. The user selects the quote they would like to view.
3. The software opens the 'Quote' window showing labels and the non-editable fields containing the quote values they describe. The quote also contains a non-editable list /table of quote items.
4. The user inspects the quote (S1) (S2).
5. The user closes the Quote window.
6. The software returns the user to the 'Quote Manager' window.

Subordinate Flows (S)

S1

1. The user selects a quote item and chooses to edit or remove it (the same way as a job).
2. The quote item list/table in the Quote window is updated or reduced.

S2

1. The user chooses to edit, remove or add a quote (see Issue Quote) - similar to a job.
2. The list of quotes in the 'Quote Manager' window is updated or reduced.

Alternate/Exceptional Flows (E)

E1

1. The user selects a job from the jobs list and chooses to view the quotes for the job.
2. The software opens the 'Quote Manager' window. (N2)

**Use Case Name: Compile Site Report**

Assumption: The user has the 'Job Manager' window open.

Brief: The user wants to produce a site report showing a complete report on the job.

Normal Flows (N)

1. The user selects a job from the jobs list and chooses the ‘create a site’ report.
2. The software opens a new window, gathers all the related information to the job, formats it and presents it to the user.
3. The user closes the site report window.
4. The software returns to the 'Job Manager' window.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: Estimate Work Hours**

Extends: Calculate Quote

Brief: The contents for the quote is being determined.

Normal Flows (N)

1. The software gets the estimated time from the quote object.
2. The software places the time into a non-editable field on the Quote window.
3. The user can see the budgeted work hours as part of the quote.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: Manage Finances**

Assumption: The user has the main window open.

Brief: The user wishes to manage expenses (like tools, materials and other business expenses) by adding, editing and removing expenses.

Normal Flows (N)

1. The user chooses to open the 'Expense Manager' window.
2. The software opens the window and loads it with a list of all the expenses. There are similar options to the 'Job Manager' window.
3. The user selects an expense and chooses to view it.
4. The software opens an 'Expense' window showing the data stored on the expense, formatted. There are similar options to when viewing a job.
5. The user closes the window.
6. The software returns the user to the 'Expense Manager' window.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

**Use Case Name: Calculate Expenses - for a job**

Brief: Essentially a part of calculating an invoice.

Normal Flows (N)

1. The software gathers all expenses related to the job and adds them to a list.
2. The software goes through the list.
3. For each item the total expense is calculated and added to a total.
4. The software returns the total to the parent function.

**Use Case Name: Calculate Expenses - for a period of time**

Brief: The user selects a period of time for which to calculate expenses.

Normal Flows (N)

1. The user chooses to calculate expenses for a time period.
2. The software shows an input dialog for entering/selecting a time period.
3. The user enters/selects a time period and accepts it.
4. The software retrieves all the expenses for that time period and adds them to a list.
5. For each item in the list a total is calculated and added to the total.
6. The software displays the total.

**Use Case Name: Add Expense**

Brief: The user wants to add an expense to a specific job, or as a business expense.

Normal Flows (N)

1. The user has the 'Expense Manager'' window open. (E1)
2. The user chooses to create a new expense.
3. The software opens a data entry window.
4. The user enters data for the expense.
5. The software validates the data.
6. The user accepts it.
7. The software adds the new expense to the database.

Subordinate Flows (S)

Alternate/Exceptional Flows (E)

E1

1. The user has the 'Job Manager' window open.
2. The user chooses to add an expense for the job.
3. The software opens a data entry window for a new job expense.
4. The software automatically links the job to the expense.
5. Go to N3.

**Use Case Name: Manage Clients**

Assumption: The user has the main window open.

Brief: The user wants to manage by adding, editing and removing them.

Normal Flows (N)

1. The user chooses to open the 'Client Manager' window.
2. The software opens the window and loads it with a list of all clients. There are similar options to the 'Job Manager' window.
3. The user selects a client and chooses to view it.
4. The software opens a 'Client' window showing the data stored for the client, formatted. There are similar options to when viewing a job.
5. The user closes the window.
6. The software returns the user to the 'Client Manager' window.

**Use Case Name: Create Client**

Brief: The user wants to create a new client.

Normal Flows (N)

1. The user has the 'Client Manager'' window open.
2. The user chooses to create a new client.
3. The software opens a data entry window.
4. The user enters data for the client.
5. The software validates the data.
6. The user selects an address for the client (S1) or adds a new one (S2).
7. The user accepts it.
8. The software adds the new client to the database.

Subordinate Flows (S)

S1

1. The user selects an existing address from a list to assign to the client.
2. The software adds the address as linked to the client.

S2

1. The software opens a data entry window for a new address.
2. The user enters the data.
3. The software validates the data.
4. The user accepts it.
5. The software adds the address to the database and links it to the client.

Alternate/Exceptional Flows (E)

**Use Case Name: Change Client Data**

Assumption: The user has the 'Manage Clients' view open and has selected the client they want to change the data of.

Brief: The user selects a client to possibly alter all of the data set when it was created.

Normal Flows (N)

1. The user chooses to edit the client.
2. The software opens a 'Client' window with data entry fields containing the current data for the selected client.
3. The user makes valid changes to the data (E1) (S1).
4. The software validates the changes.
5. The user accepts the changes.
6. The software updates the database.
7. The data entry window is closed.
8. The client list on the 'Client Manager' window is updated with the altered client.

Subordinate Flows (S)

S1 - The user chooses to select a different address (S2) or alter an existing one (S3).

S2

1. The software presents a list of all existing addresses.
2. The user selects a new one from the list.
3. The software links the address to the client.

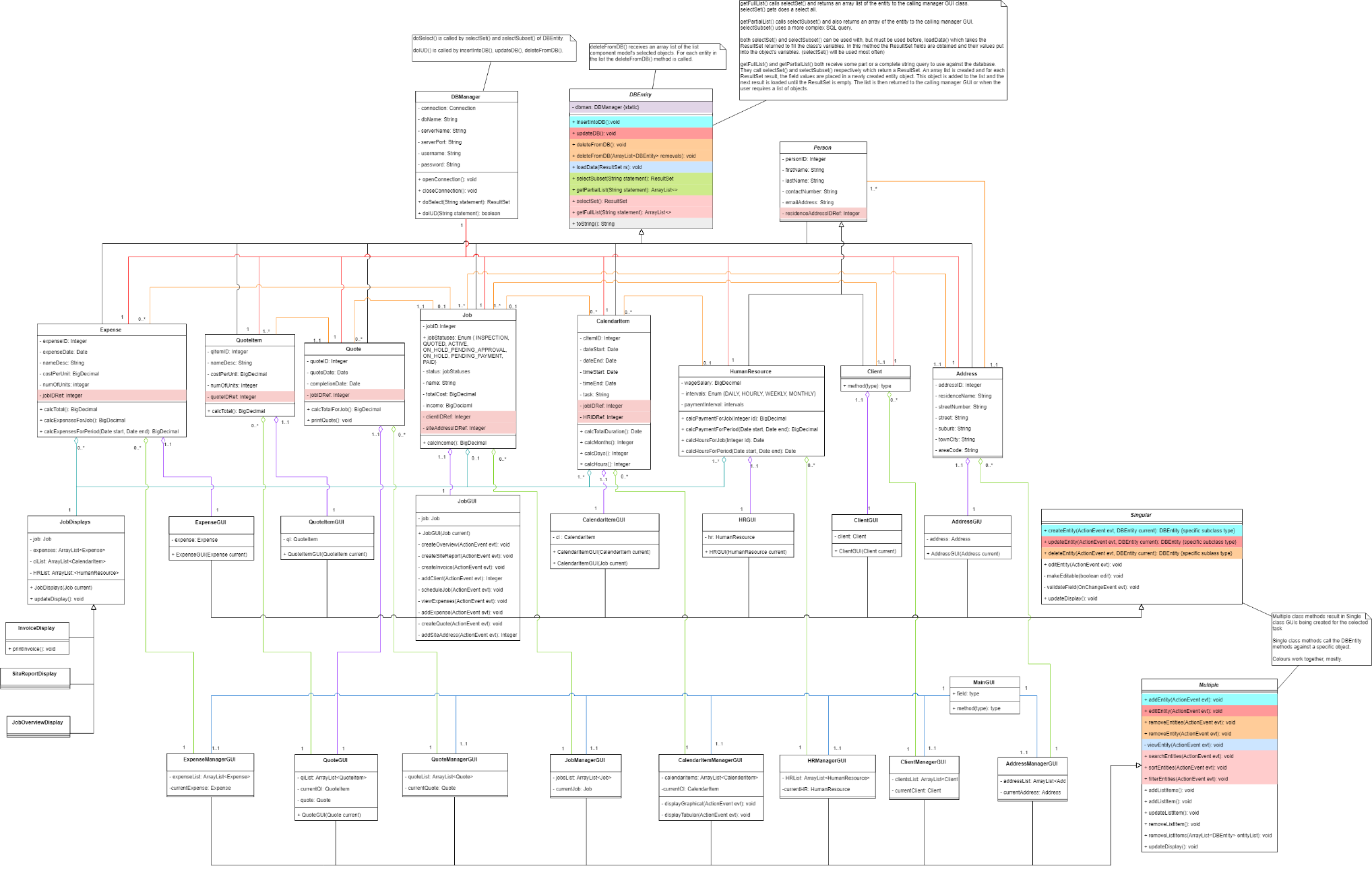
S3

1. The software opens a new 'Address' data entry window with the fields containing the existing data.
2. The user makes valid changes to the data (E1).
3. The software validates the changes.
4. The user accepts the changes.
5. The software updates the database and closes the 'Address' window.

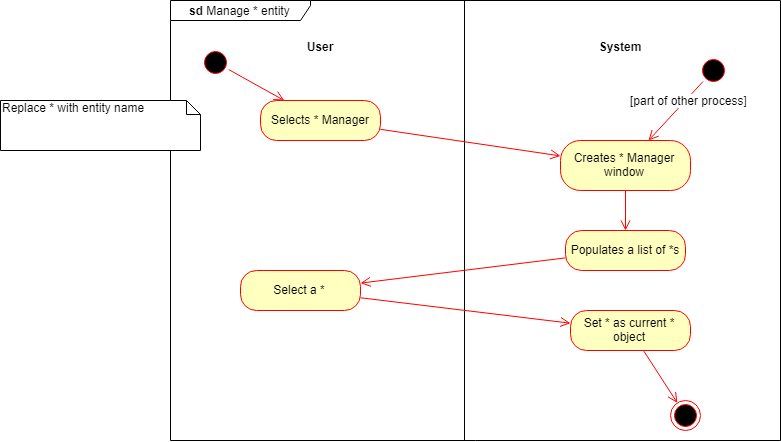
Alternate/Exceptional Flows (E)

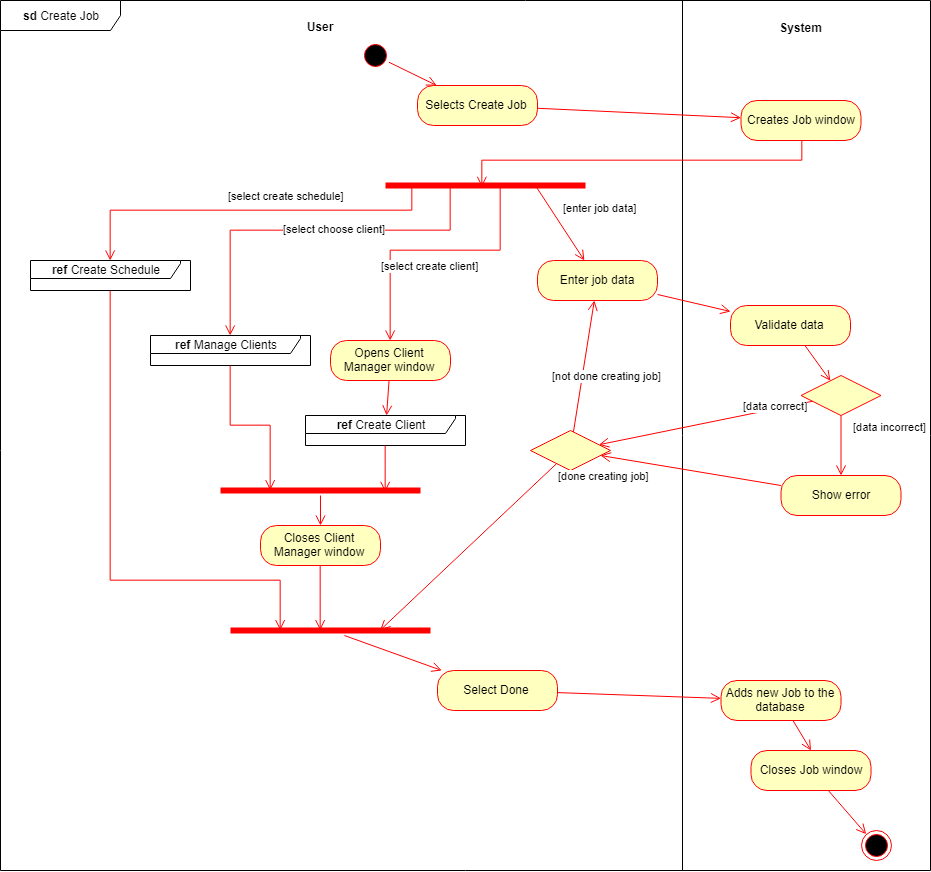
E1

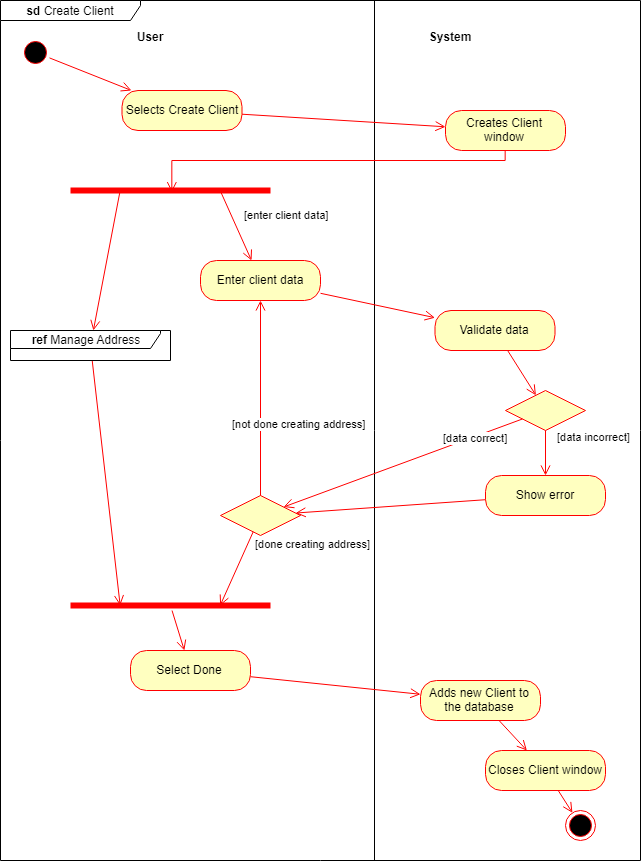
1. The user enters incorrectly formatted or typed data.
2. The software indicates this to the user.
3. The user makes corrections.
4. The software validates the entered data again.
   * 1. Class diagrams

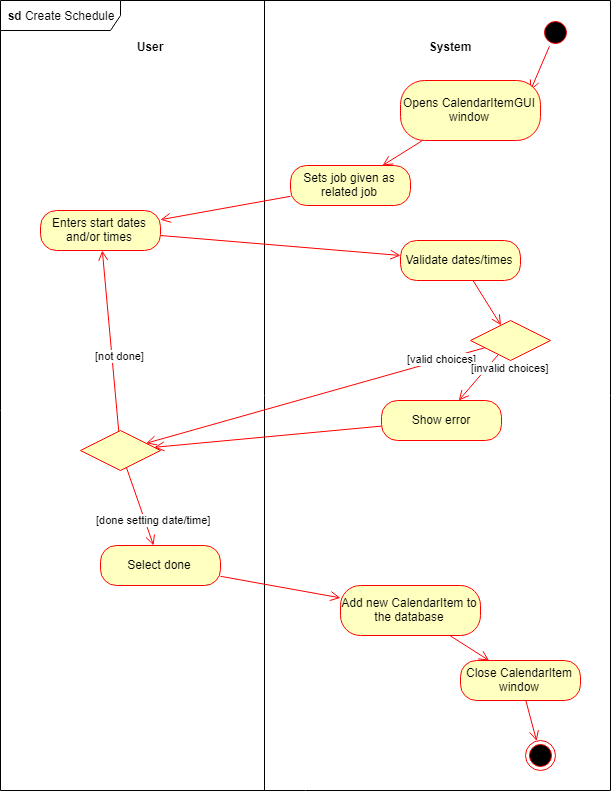


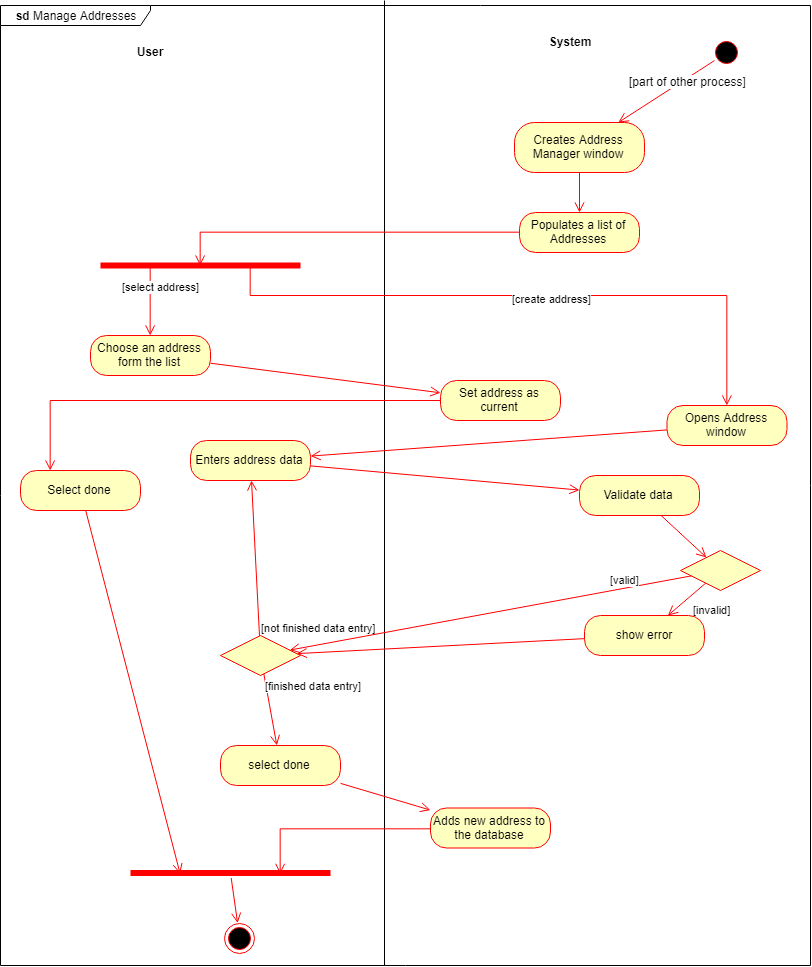
* + 1. Activity diagrams



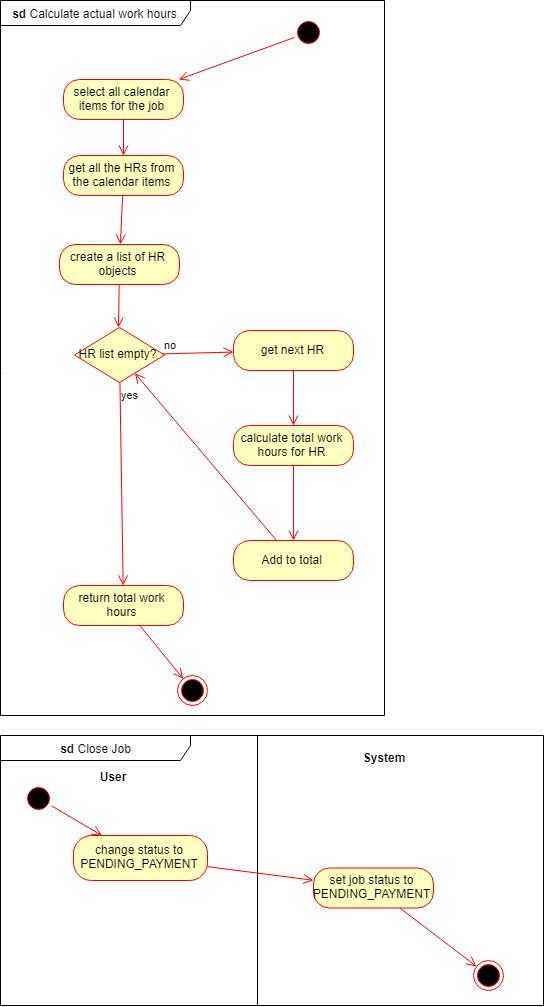


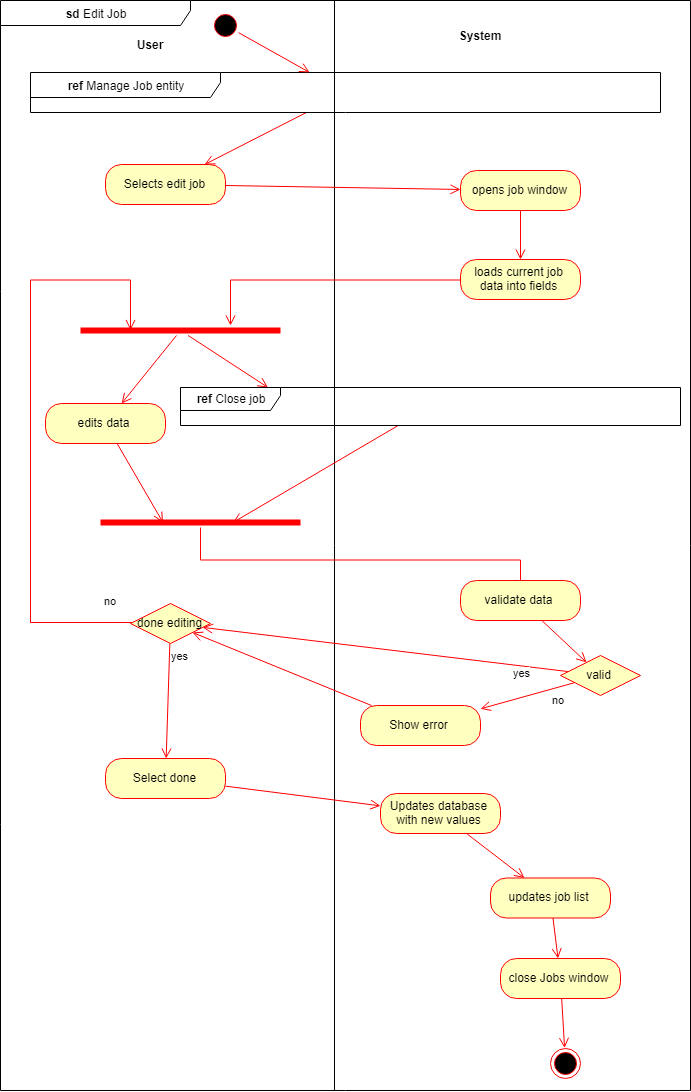


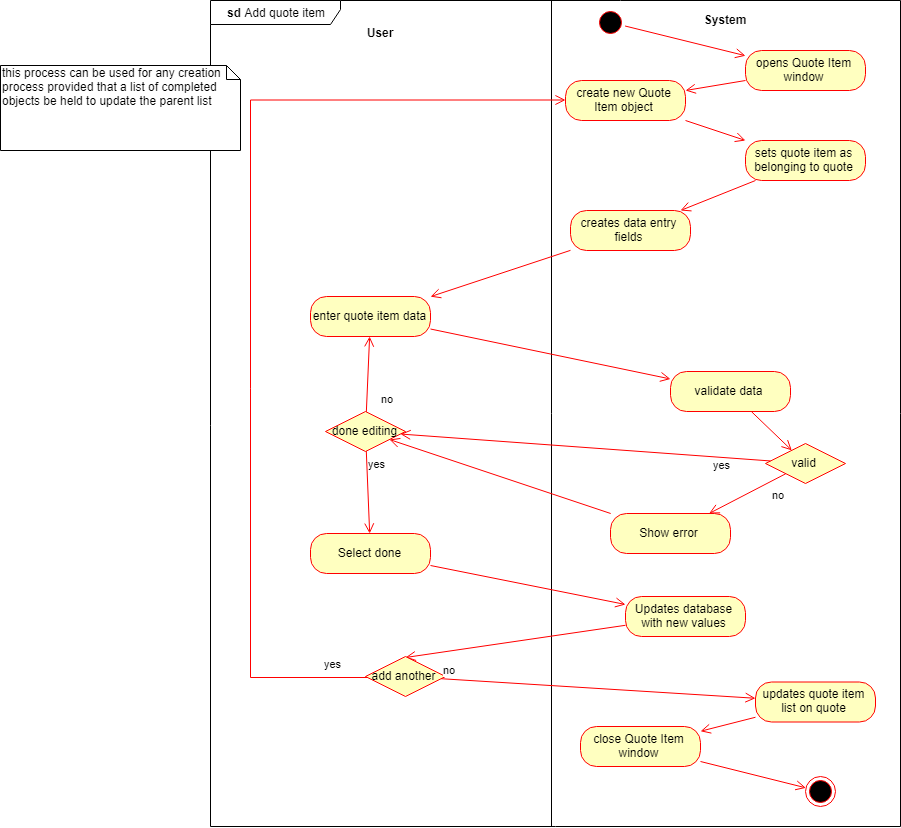


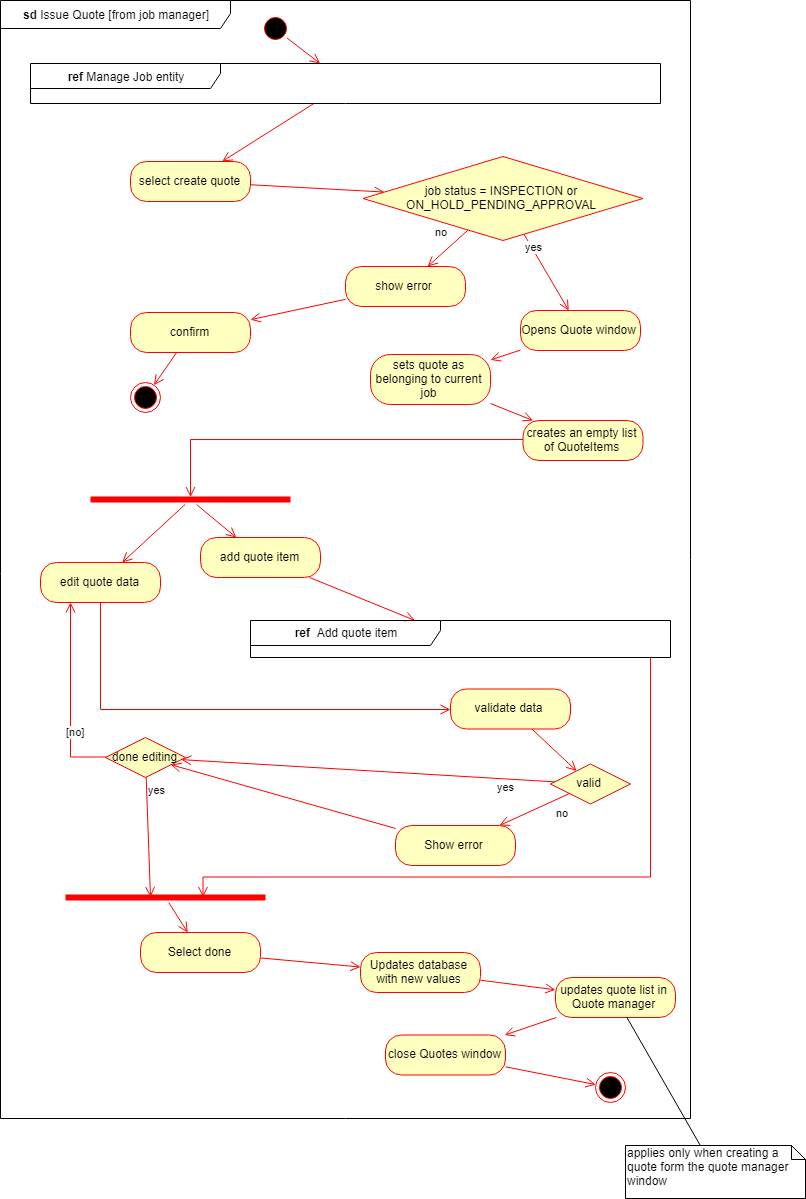


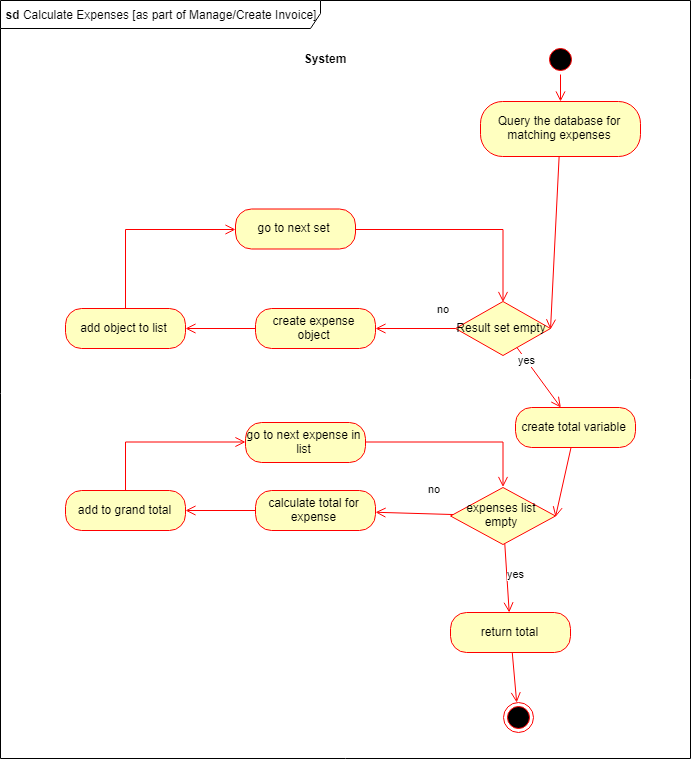


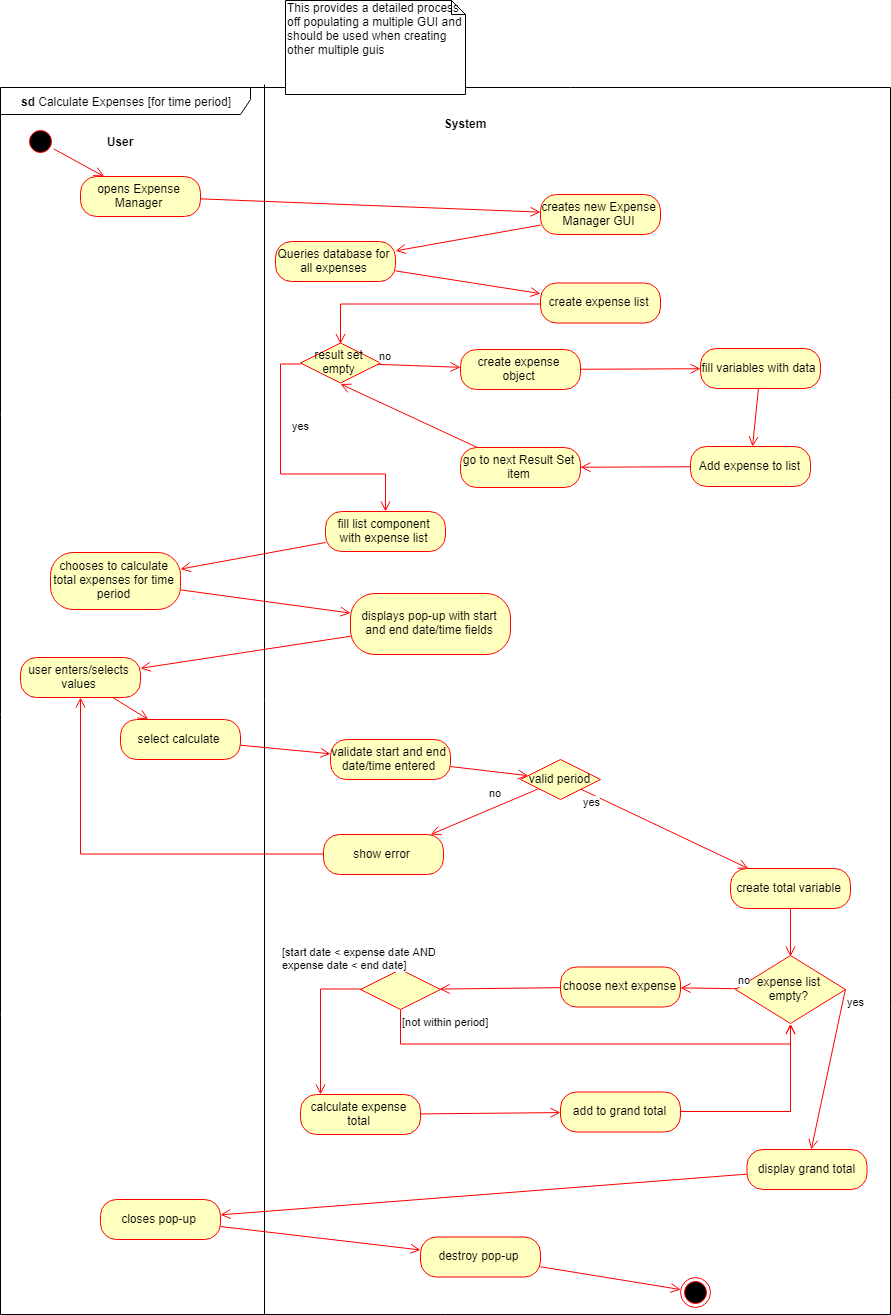




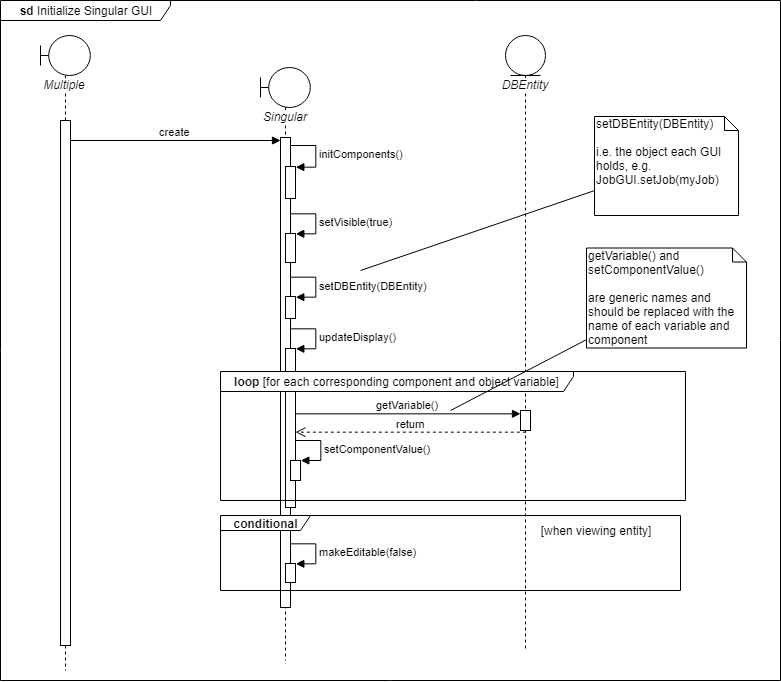


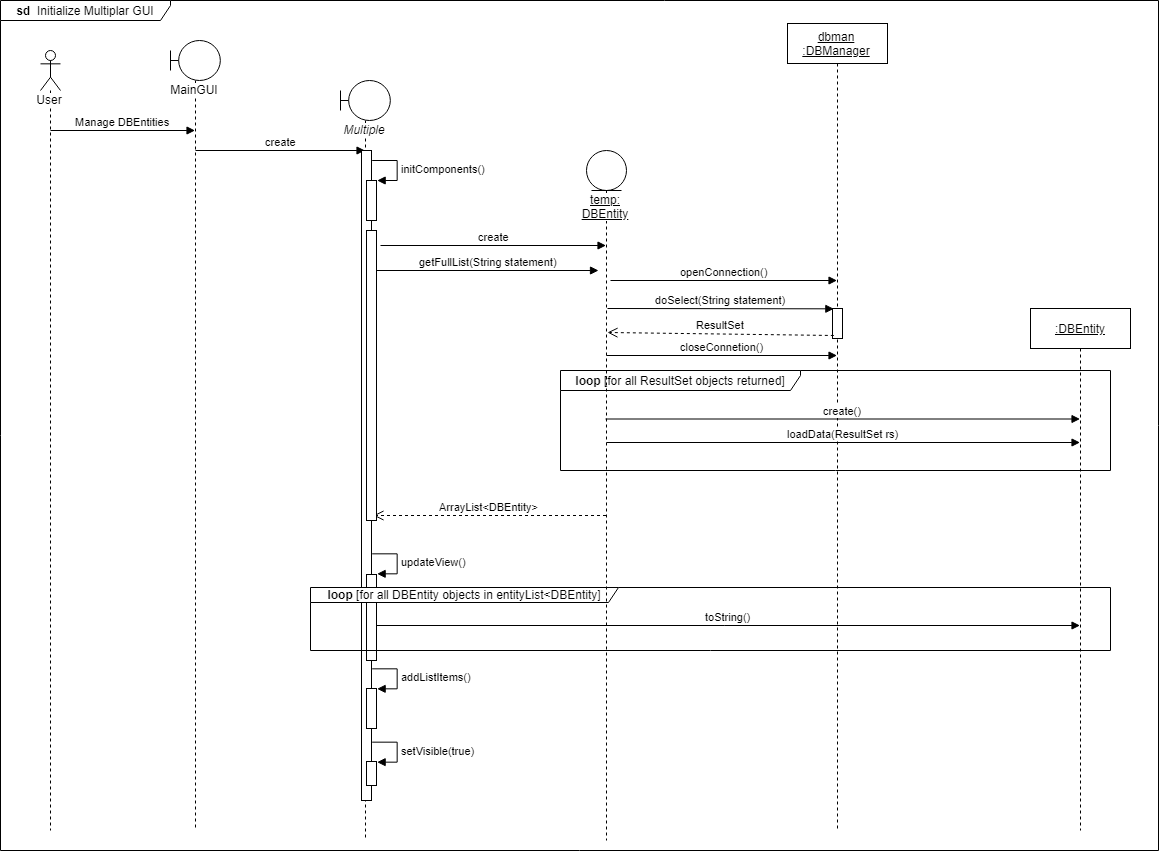


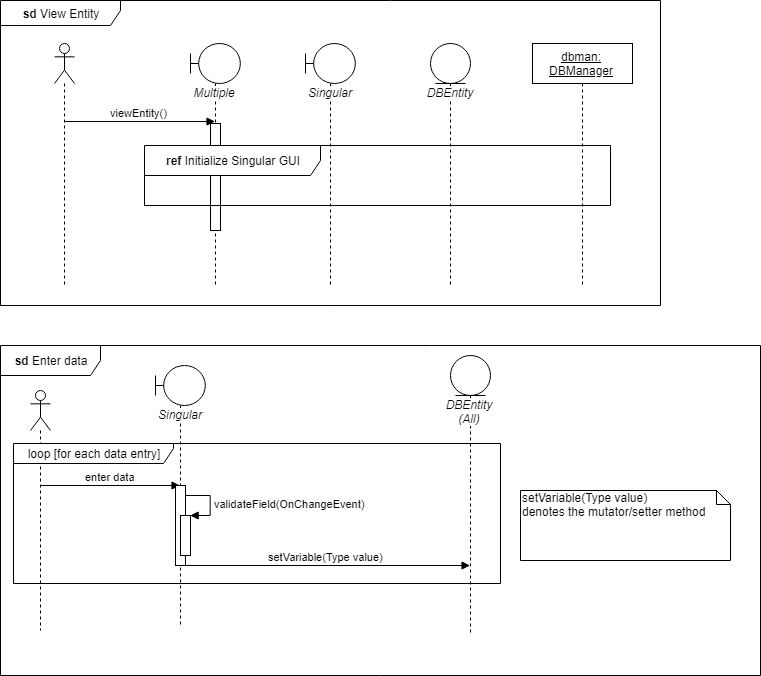


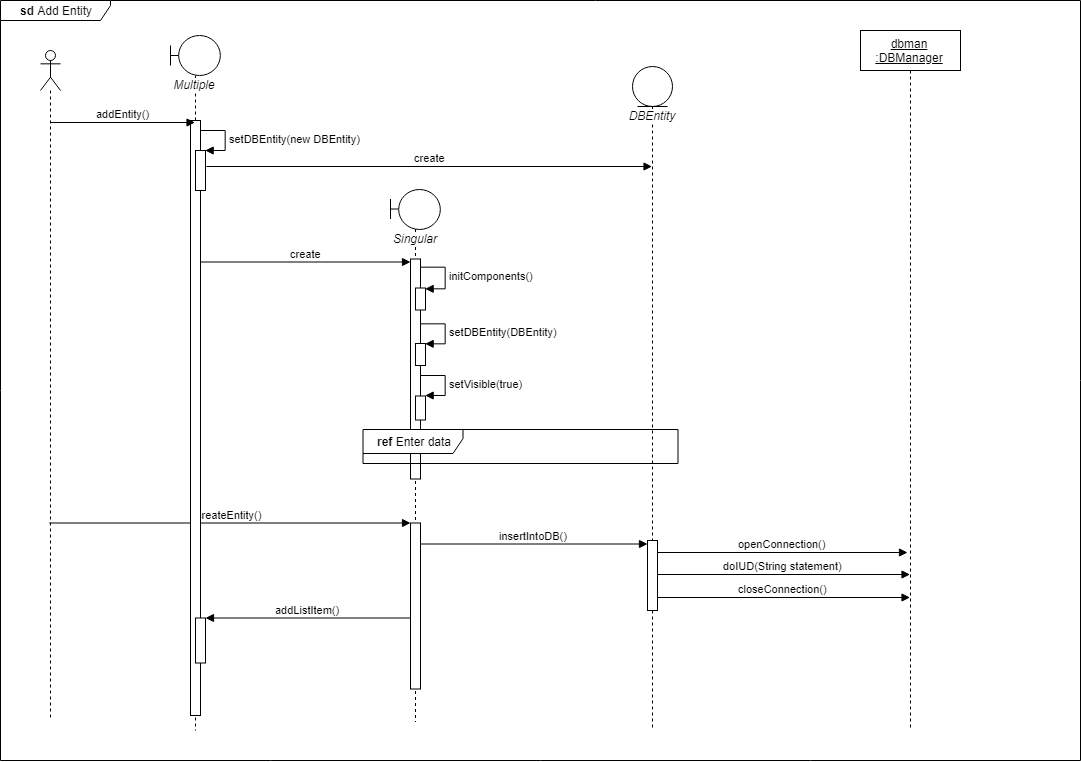


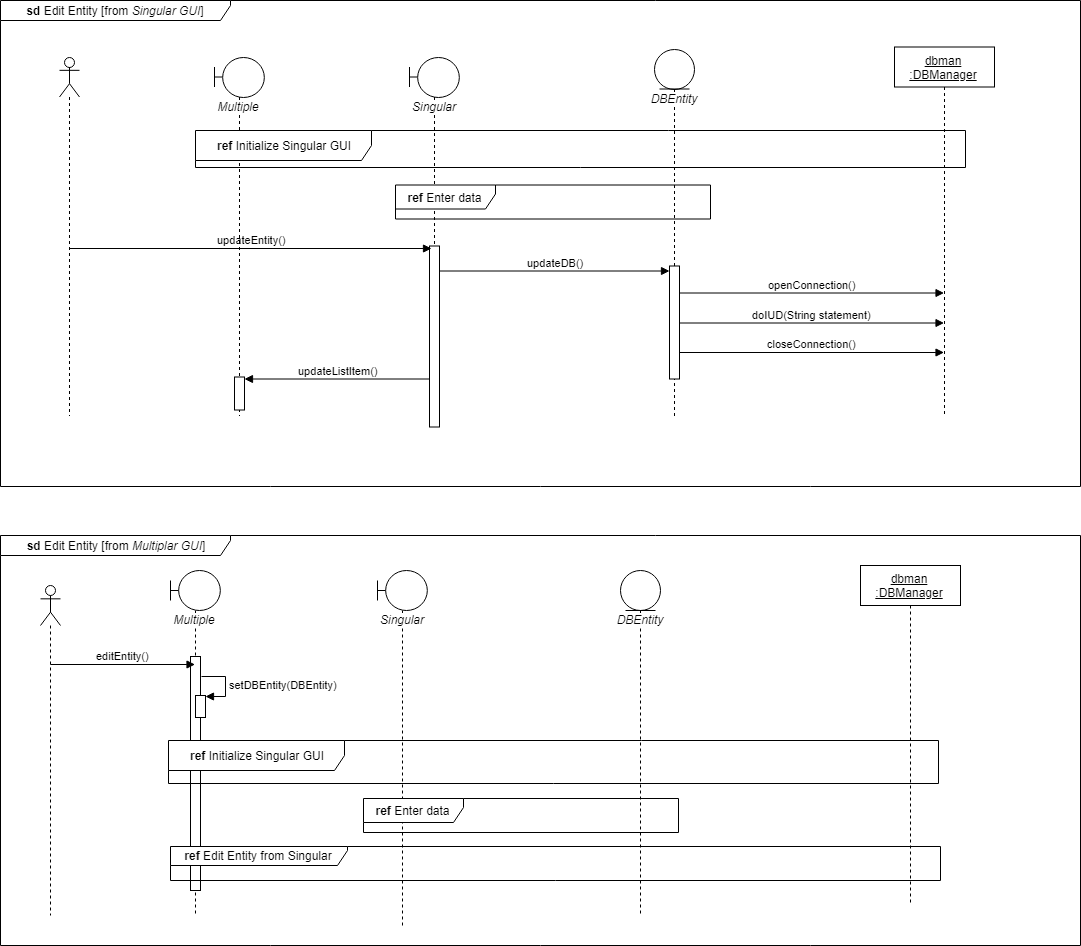
* + 1. Sequence diagrams

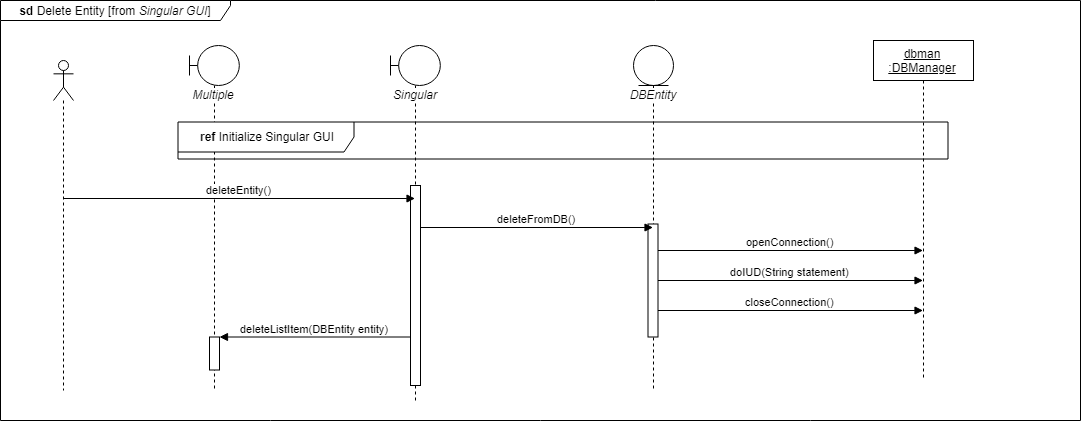


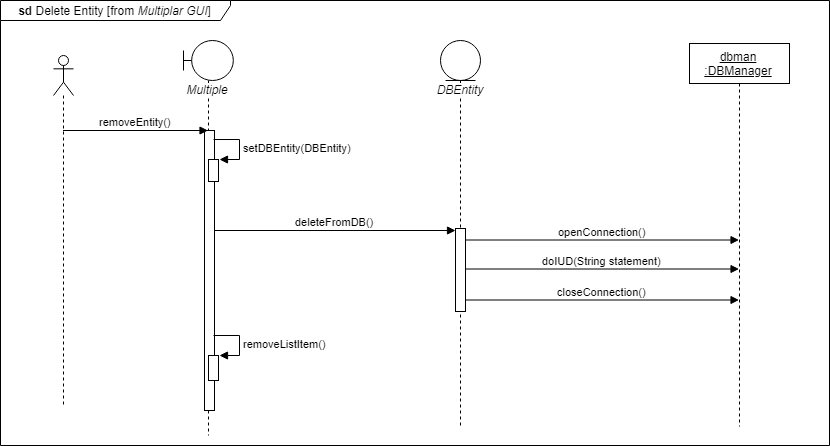


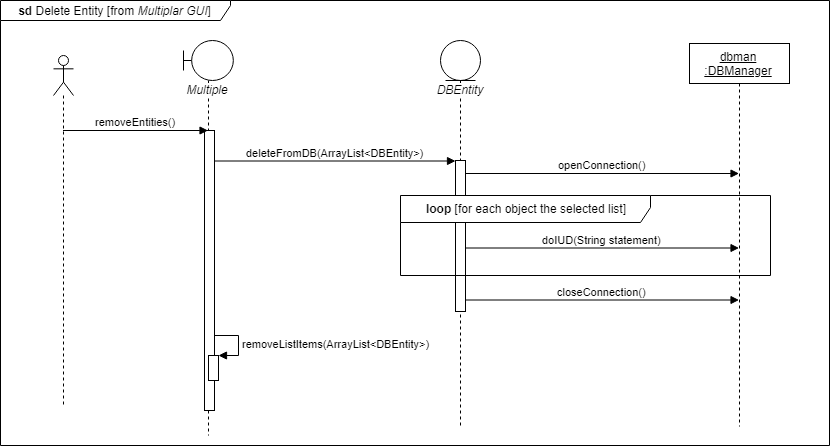












* 1. Physical design
     1. Investigation of technologies to be applied

**Operating System (OS)**

**Microsoft Windows 10**

This OS was chosen as the main development platform as it is the industry standard, Microsoft Windows 10 Enterprise is the latest instalment of the Microsoft Windows franchise and is well known by the development team as well as the client.

**Programming Languages**

**Java**

The client specified a desktop application as the preferred type of system, Java is an excellent language for the development of desktop applications and was selected by the development team, because of their high level of expertise in the programming language. The system was developed using the latest available version, Java Version 8 Update 181.

**Structured Query Language (SQL)**

As data has to be stored and retrieved from a central database, SQL has been selected as the query language. SQL is also an industry standard.

**Integrated Development Environments (IDE)**

**NetBeans**

NetBeans is a free to use IDE developed by Sun Microsystems (since acquired by Oracle Corporation). It is mainly used for educational purposes and smaller projects, but was chosen as it is fitted with a Graphical User Interface (GUI) Builder, which simplifies the design process. The system was developed using the latest available version, NetBeans IDE 8.2. It is also important to note that NetBeans makes use of the Java Development Kit (JDK), and for this system JDK 8u181 was used.

**Microsoft Access**

Access is a Database Management System (DBMS) developed as part of the Microsoft Office instalment. It is a lightweight DBMS that does not require the database to be hosted, which is perfect for desktop applications. Microsoft Access 2013 was chosen to develop the database. It is also important to note that to ensure connectivity between the Java system and the Access database, a Java Database Connectivity (JDBC) driver is required. UCanAccess was chosen as this driver.

**Packaging and Deployment**

**Launch4j**

This program was used to package the Netbeans project into an executable .exe file, for ease of use by the client. Launch4j Executable Wrapper is a free to use, lightweight tool.

**Inno Setup**

This program was used to deploy the executable file into a setup file, for ease of use by the client. Inno Setup 5.6.1 is a free to use, lightweight tool.

* + 1. System testing

**Testing Types**

Testing makes up a very important part of software development and is performed throughout the development process especially when the programmer is coding the software however there is also a special phase with in the development life cycle set out for testing. Testing was initially done to ensure that developed software functions properly but have since developed to be to be used for a range of reasons, which includes ensuring a good user experience, the software developed is what the client expects or wants, ensuring the software runs efficiently and effectively. The list of reasons is long. Just as the reason for testing have evolved and multiplied over the years so have the test being performed. Today there are various test that can be performed on software making the testing process much more effective and efficient ensuring high quality software is produced. Today software testing is divided into two categories which is blackbox testing and whitebox testing. This is the basic forms of testing software.

Blackbox testing means that the tester runs the software as a user and observe the software to investigate if the software runs as expected. This can be done by the tester have a set of data that includes the initial data and the expected results. The tester will feed the initial data into the software and evaluate if the software yields the expected result. The tester is expected to report on their findings. This type of testing has advantages:

* Like the tester does not need any coding knowledge or experience.
* Less time consuming as the tester does not need to work through any code.
* The tester work independently from programmers or designers making the tester unbiased.
* Test can be developed once requirements is finalised.

Blackbox also has disadvantages like:

* Complex segments of code cannot be tested using blackbox testing.
* Often leaves testing paths untested as a result of the quantity of test paths.
* Many of the test ran may already have been executed by programmer.

There are various form of blackbox testing that can be implemented like:

* Boundary Value Analysis
* Equivalence Class Partitioning
* Decision Table Based Testing
* Cause Effect Graphing Technique
* Error Guessing

Whitebox testing entails working through the code of the software to ensure the software works correctly. This type of whitebox testing is known as a walk through and is the most popular forms of whitebox testing. It can mean the tester running the code and if the expected result is not yield the tester will inspect the code line by line to locate and correct the bug in the code. Whitebox testing has advantages like:

* Produce optimised software as all hidden bugs are eliminated.
* Effectively test software as it test all possible testing paths.
* Test are updated each time the software is modified for any reason.
* Provides helpful feedback to the programmers and designers helping solve the errors.

Whitebox testing also have disadvantages like:

* Requires a certain level of experience and understanding which turns out to be expensive and complex.
* Is an ongoing process which happens concurrently while software is being developed as test needs to be updated each time a change is made.
* Become time consuming when each testing path is tested and often result in not all testing paths is tested.

Other forms of software testing includes but is not limited to:

* Unit Testing
* Integration Testing
* Functional Testing
* System Testing
* Stress Testing
* Performance Testing
* Usability Testing
* Acceptance Testing
* Regression Testing
* Beta Testing

**Testing Template**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | |  | |
| **Role Of Tester:** | |  | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** |  | **Date Tested:** |  |
| **Software Used:** |  | **Test Type:** |  |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
|  |  |  |  |
| **Result** | | | |
| **Test Pass:** |  | **Recommendation** |  |
| **Screenshot:** |  | | |
| **Comments** |  | | |

**Test plan**

The test plan consist of all the details on how the tests will be executed. This entails what test will be used in the test, at what stage these tests will be executed, who will be executing these tests, what the outcome of the test and conveys any comment the tester wants to document.

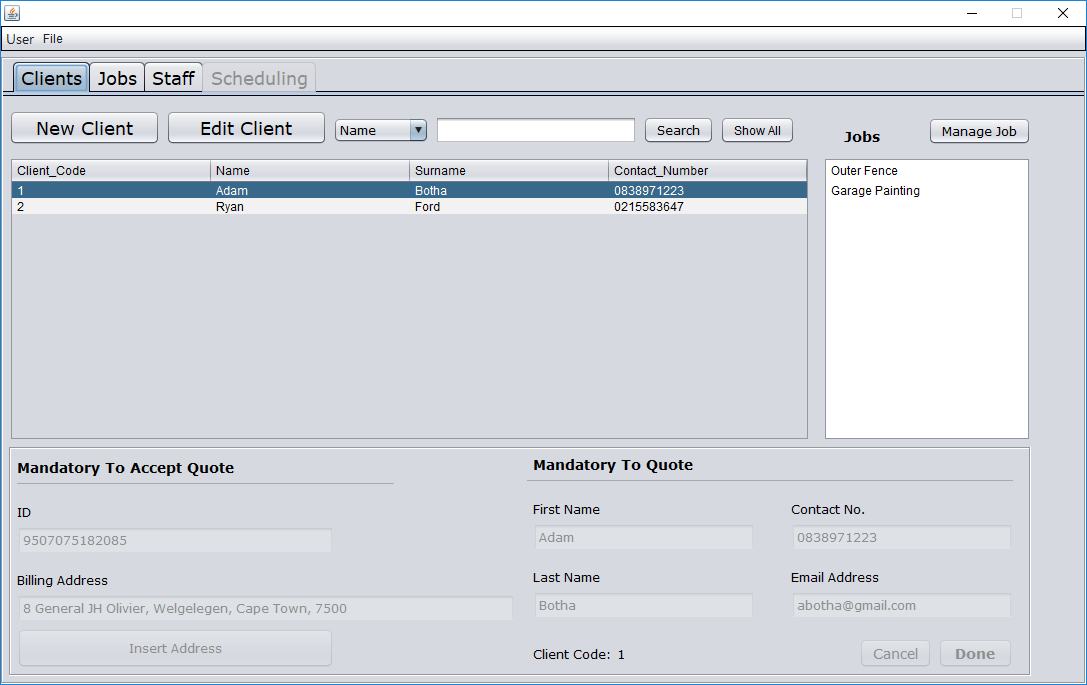
The test that will be used in the test plan in question will be black box to cover every major area of the software developed. Once a problem is flagged during the black box testing it will be further investigated by using white box testing to ensure that the testing of the software benefit from both the advantages of the type of testing and avoid the disadvantage of the types of testing incorporated in the test plan.

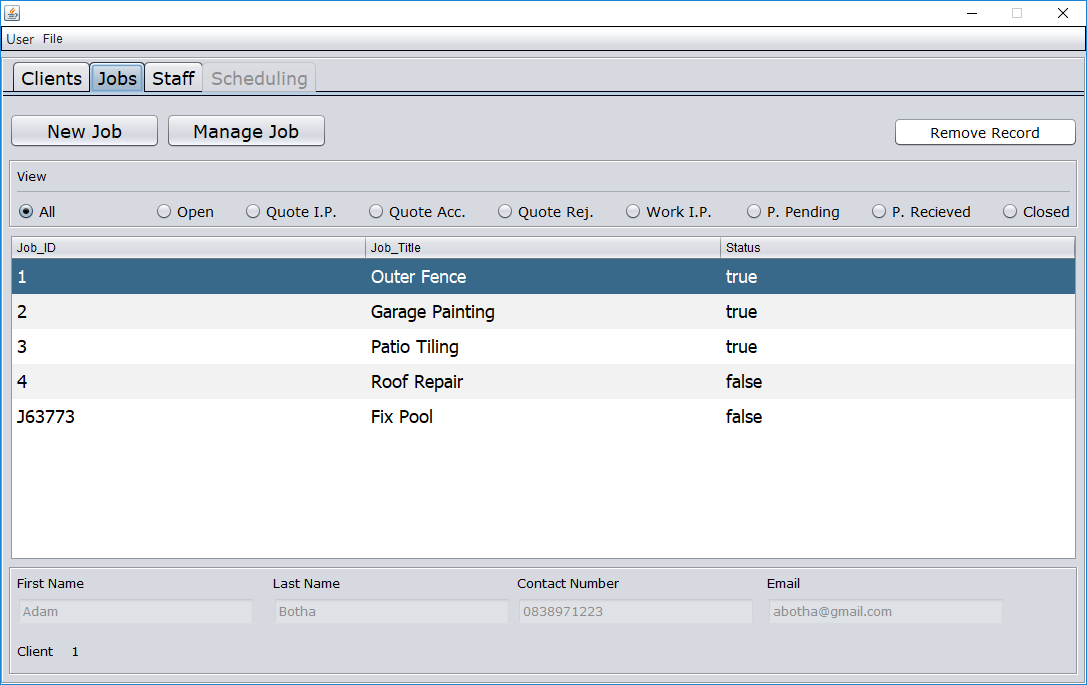
The test will be executed once each iteration to ensure that all the functions and features of the software works. All the software that was developed in the previous iterations will be retested with every following iteration to ensure it integration with the newly developed software.

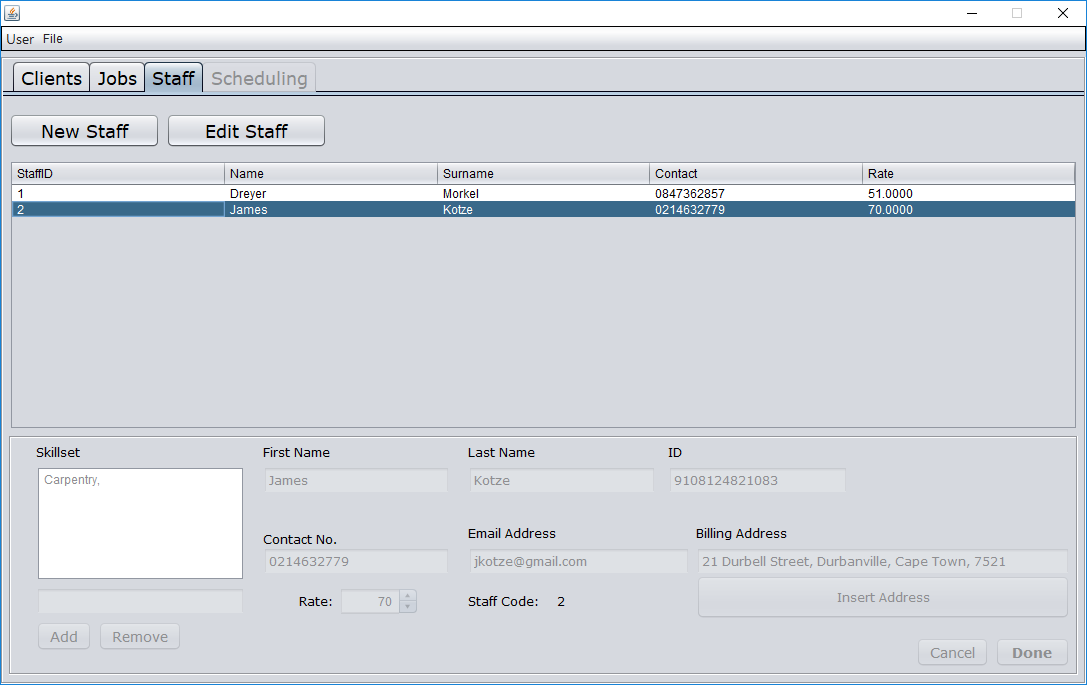
Thus the test plan will serve as a method of planning what will happen to ensure that everyone know what needs to happen when and how it should happen. It also serves as a summary of what happened in the tests. Therefore documenting everything happening for future reference.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Plan | | | | | |
| Increment | Test Type | Test Date | Team Member | Successful | Comments |
| 1 | Black box testing (Client Page) | 02 September 2018 | Jonathan Prince | Fail | Ensure a client can be added before working on additional features. |
| 1 | Black box testing (Job Page) | 02 September 2018 | Jonathan Prince | Pass | None. |
| 1 | Black box testing (Staff Page) | 02 September 2018 | Jonathan Prince | Fail | Work on adding a staff member first. |
| 2 | Black box testing (New/Manage Job Page) | 20 September 2018 | Jonathan Prince | Fail | None. |
| 3 | Black box testing (Quote Page) | 24 October 2018 | Jonathan Prince | Fail | Ensure that a quote can be created first successfully |
| 4 | Black box testing (Work Page) | 31 October 2018 | Jonathan Prince | Fail | Aim to get work page functional before doing any other work on page |
| 5 | Black box testing (Finalise Job Page) | 31 October 2018 | Jonathan Prince | Fail | Check the connection to the database |

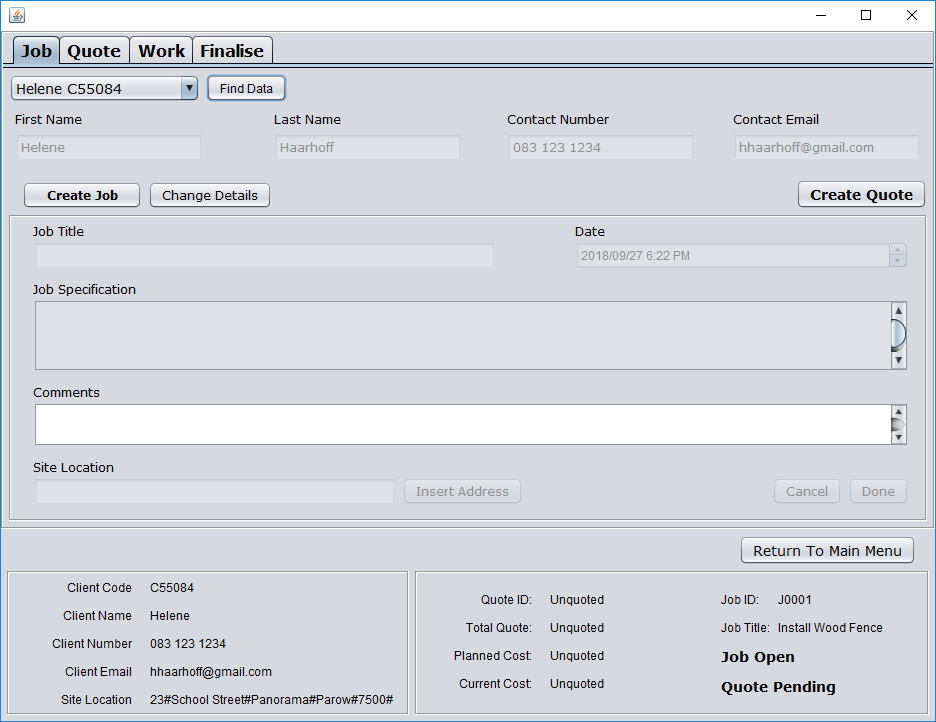
1. Deliverable 4: Implementation
   1. System interface design
      1. Iteration 1: Landing Page



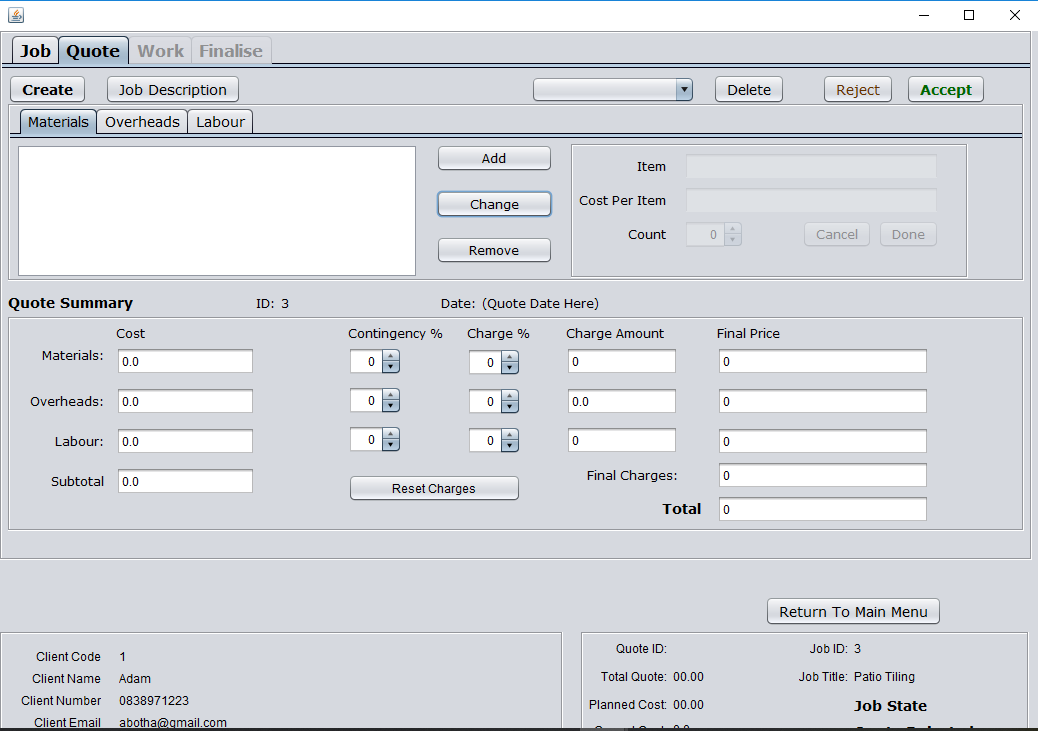




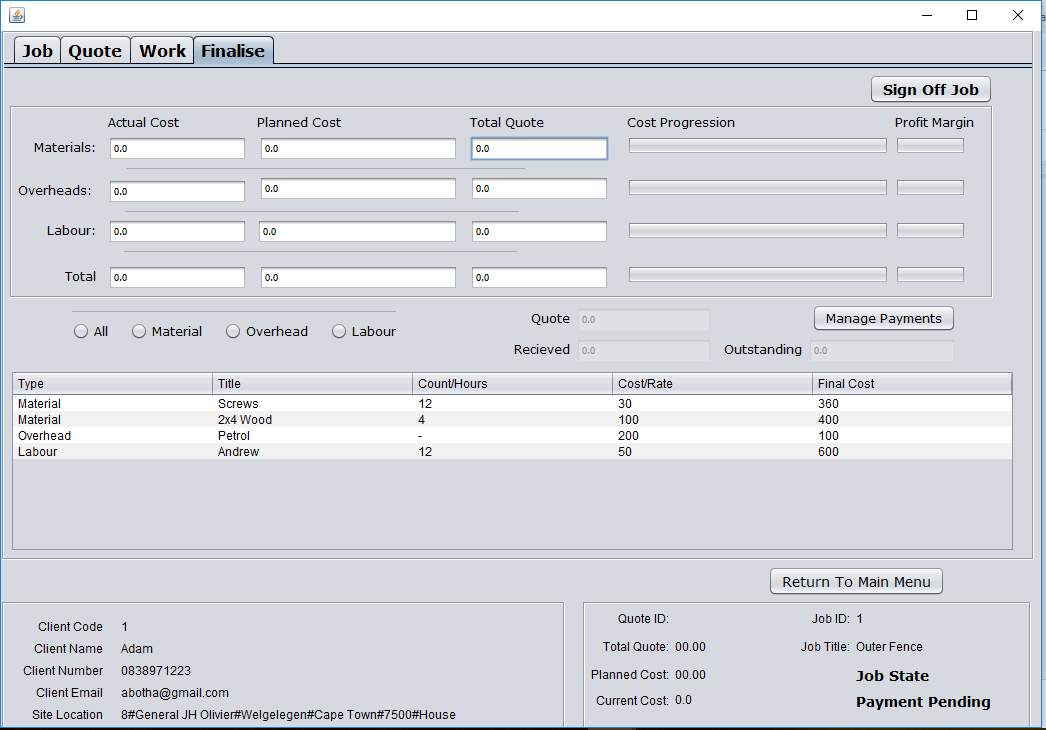
* + 1. Iteration 2: New/Manage Job Page



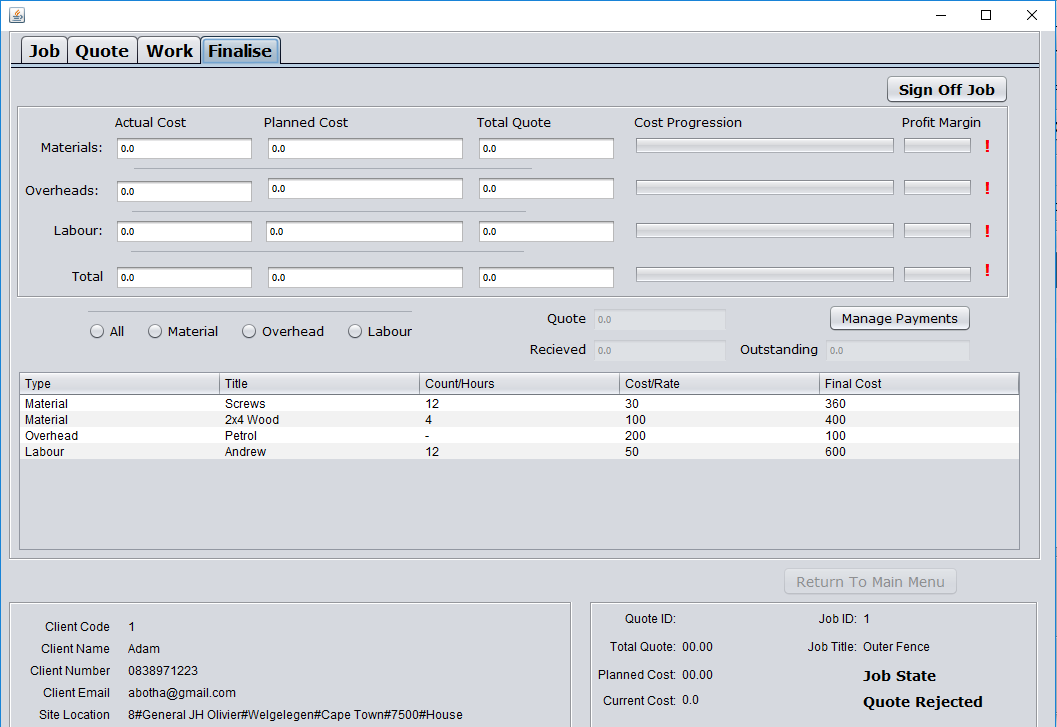
* + 1. Iteration 3: Quote Page



* + 1. Iteration 4: Work Page



* + 1. Iteration 5: Finalise Page



Testing results

* + 1. Iteration 1: Landing Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Tester** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | Client Tab | **Date Tested:** | 02-09-18 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Netbeans Opened, Database located and connected | Run the project and do a walk-through of the GUI | Project to run and all the buttons to execute as per their labels. | User button right at the top does not work.  Save and Close buttons at the top of the page are redundant.  Edit client needs to validate that a client is selected.  Search function does not show the results in the list  The whole Jobs section needs to be disabled when a client is not selected or client does not have any jobs.  Manage Job should only be enabled once a job is selected.  List of clients must be greyed out once a client is being added or edited.  When submitting a new client an error occur.  Unable to submit an edited client.  Remove Insert Address button |
| **Result** | | | |
| **Test Pass:** | Fail | **Recommendation** | Ensure a client can be added before working on additional features. |
| **Screenshot:** | https://lh4.googleusercontent.com/FXpaH_HsKRP6mA5iwMbdn7CkDCAUfGDtf4C1pcxsOj2rs3AnEmXYQDEYALNI168RksvCeYvVqvqHK252dL3gnegcfRSUnD3DYqiNRVuWTaFsBkwWY4JW5ODt3z-re7HW8gIydY9E | | |
| **Comments** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Tester** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | Job Tab | **Date Tested:** | 02-09-18 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Netbeans running and database connected | Run the project and walk through all the buttons on the tab. | All buttons to work as per labels. | Manage Job should be disabled until a job is selected.  Remove record should be disabled until job is selected.  All the various views work well.  Client details need to be dynamic. |
| **Result** | | | |
| **Test Pass:** | Pass | **Recommendation** | none |
| **Screenshot:** | https://lh5.googleusercontent.com/3WhxaaK_dGm3aghD6euQ0mE_TNy06pBRA5YSynsx7lxTTp8rSs2NkK1A0A9iIcaPpPiFq3NDidnip1tm8hj9aPvRzmQ0F80Ln4C7eyu5Y4L14o4U80qKrypfPz2SLnFHv8uNT4V- | | |
| **Comments** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Testing** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | Staff Tab | **Date Tested:** | 02-09-18 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Netbeans running and database connected | Run the project and walk through all the buttons on the tab. | To be able to add a new staff member.  To be able to edit existing staff members. | The done button to add a new staff member does not work.  The edit staff button should be disabled until a staff member is selected.  Add button in the add skill set section is not working  Remove the insert address button. |
| **Result** | | | |
| **Test Pass:** | Fail | **Recommendation** | Work on adding a staff member first. |
| **Screenshot:** | https://lh5.googleusercontent.com/qu5mo-2_yUH55uBFUbusqEp9YqYz2_-iVh7DXZLm6tD62GUYyq5LQut8J61DPkmMhluXrzNQ9Iyk8CSjmrEYNAccXkzw2KuSiwxBgUsMVLQAT7Je8nXLdlw1rFvYLFNdrBFPtSFE | | |
| **Comments** |  | | |

* + 1. Iteration 2: New/Manage Job Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Tester** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | New/Manage Job | **Date Tested:** | 20/09/18 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Copy the files into netbeans and set up database | Run the main GUI and execute a walk-through of the GUI. | The GUI to run smoothly while flagging no errors as well as perform as GUI says. | Manage Job button on the client Panel unreactive.  New Jobs button on the jobs panel is unreactive.  Manage Jobs Button on the Jobs panel is not disabled once a job is unselected.  Comment text field needs to be disabled.  Find Data is unreactive.  Error is flagged once job specifications is added.  Site location needs to be amended as discussed.  Remove record button unreactive. |
| **Result** | | | |
| **Test Pass:** | Failed | **Recommendation** |  |
| **Screenshot:** |  | | |
| **Comments** |  | | |

* + 1. Iteration 3: Quote Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Tester** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | Quotes Tab | **Date Tested:** | 24/10/18 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Project setup in netbean and database copied in the right position | Run the main GUI and perform a walk through of the interface on the quote tab. | The quote tab to open with no errors and to be able to compile a quote by adding a items | Once the quote tab is selected a warning is flagged.  Create button does not get disabled once a quote is being created.  Add button works well however capture changes needs to be disabled when adding.  additions is not reflected in the adjacent panel when done is clicked.  In the quote summary section none of the values are auto populated.  Capture button appear to be unresponsive due to no confirmation of capture.  client id is redundant in client details. |
| **Result** | | | |
| **Test Pass:** | Failed | **Recommendation** | ensure that a quote can be created first successfully. |
| **Screenshot:** | https://lh5.googleusercontent.com/eXr8k2HS55feYugMo1nnRsDL9hZE_Db_9KLDNFkyYPnjKiht7hT9zrAbFgGjy1PCdi8Sn2dqwt3fFWhMFaYJFb5TwnwtB0p68-hNqUwyuvUGk_9dFvATny0PLCEB33p6rxajQpRf | | |
| **Comments** |  | | |

* + 1. Iteration 4: Work Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Tester** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | Work Tab | **Date Tested:** | 31/10/2018 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box Testing |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Open and run Java file. | Navigate to the work page and updating the work page while doing a walk through of all the functions. | The work page to load as designed. The user be able to update the work done on the selected job. | The work page loads as designed. When the user adds a item to the list it clears the whole list.  Finalise Job button works as expected. |
| **Result** | | | |
| **Test Pass:** | Fail | **Recommendation** | Aim to get work page functional before doing any other work on page |
| **Screenshot:** | https://lh6.googleusercontent.com/y7v8GjsPSl-KaN_-jbdhxwMljpVcJyv9jHe-og3SoplPoDYE9NXYFEu-2jHmNwTmCt7lD8UbRhKuvtQoLFVSTQ_aos3YMYkC3Ke3lRMDr1nkCnyEc7_z1_tGh5wkZnNqArOOkO_X | | |
| **Comments** |  | | |

* + 1. Iteration 5: Finalise Job Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Tester Information** | | | |
| **Test Performed By:** | | **Jonathan Prince** | |
| **Role Of Tester:** | | **Black Box Tester** | |
| **Signature:** | |  | |
| **Test Information** | | | |
| **Unit Tested:** | Finalise Page | **Date Tested:** | 31/10/2018 |
| **Software Used:** | Netbeans | **Test Type:** | Black Box |
| **Test** | | | |
| **Initiation Conditions** | **Test Performed** | **Expected Outcome** | **Result** |
| Copy database into correct place. Open netbeans and open project. | Run the main GUI and navigate to the finalise page. Check the functionality by doing a walk through of the page. | The page to load as designed.The usage of the job to be broken down in the top section. The radio buttons to show only the selected type of expenses on the job. To enable the user to add payments and sign off the job. | The usage is not populated per what was used in the job.  The radio buttons is unreactive to selections.The user is able to add a payment, however the payment is not reflected on the page. The user is able to sign off a job. |
| **Result** | | | |
| **Test Pass:** | Fail | **Recommendation** | Check the connection to the database |
| **Screenshot:** | https://lh4.googleusercontent.com/Vk882OWfXtwaVo2QRWtEF373rVd7vi9jQXxMnN5oTrpkTo5sKYdsAmVPumT5mZYjHpuDWT-Pj1Zbr9FDEpTIKMU8OMkLq7_XoaJwPSFeyLdkSbvtGngQufeKrpiZ9X0T25On8mal | | |
| **Comments** |  | | |

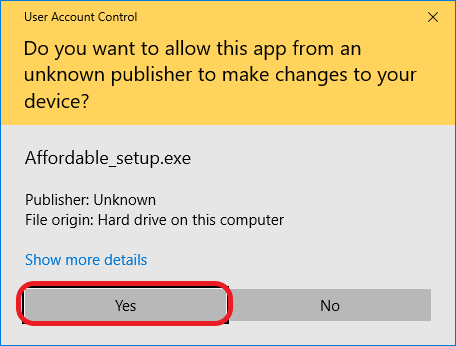
1. Deliverable 5: User manual
   1. Introduction

Affordable MIS is a Management Information System (MIS) specially created for Mr Ricky Ford. Mr Ford is the owner of a home maintenance and repair company situated and operating in Bloubergstrand. Affordable MIS will assist Mr Ford in the management of clients, jobs and staff members. Affordable MIS will further assist Mr Ford in the creation of quotes, keeping track of expenses & work being done, as well as the management of various payment options. All information will be stored and the progress of jobs will be tracked for reporting purposes.

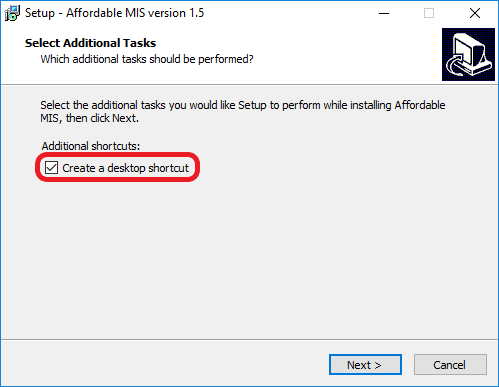
* 1. How to install
     1. Double click on the Affordable MIS setup file



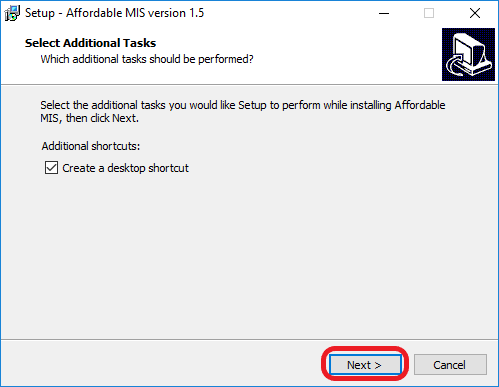
* + 1. Select Yes



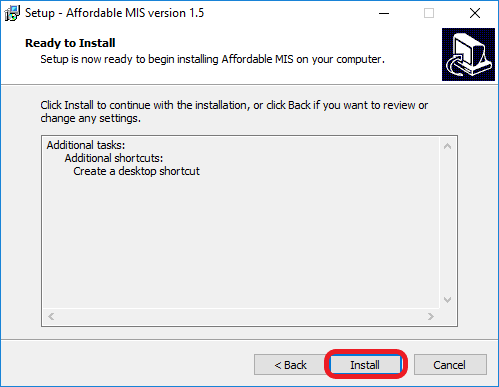
* + 1. Select Create a desktop shortcut



* + 1. Select Next >



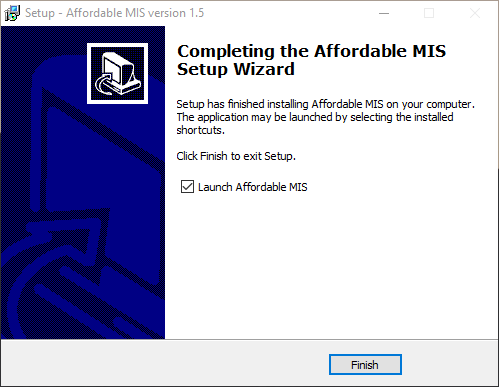
* + 1. Select Install



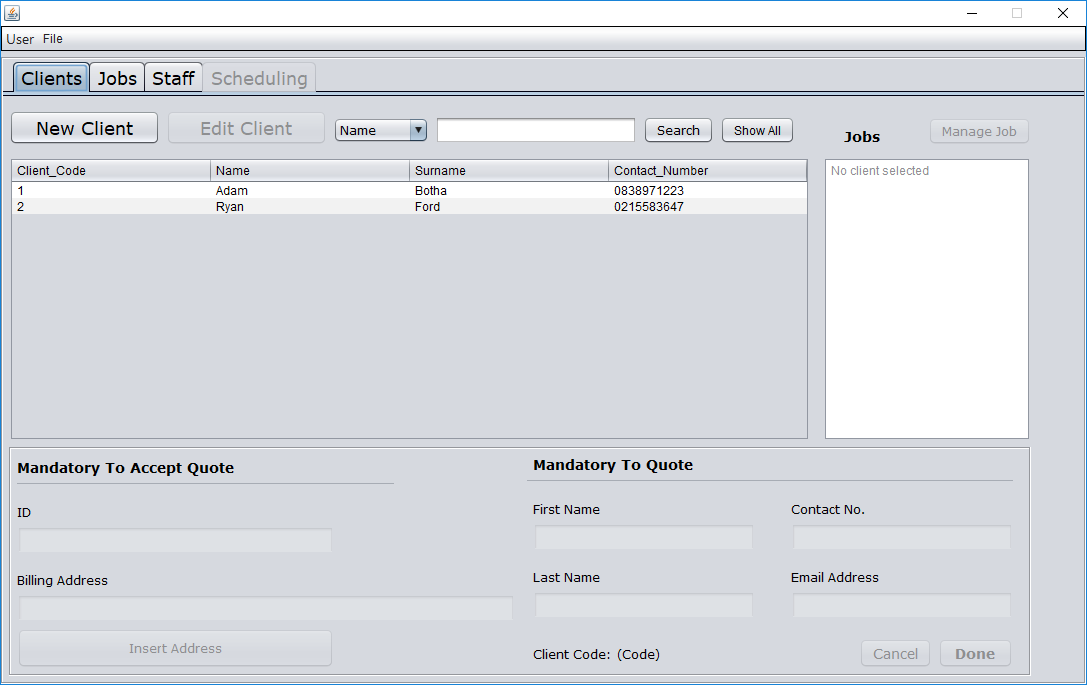
* + 1. Wait for setup to install application



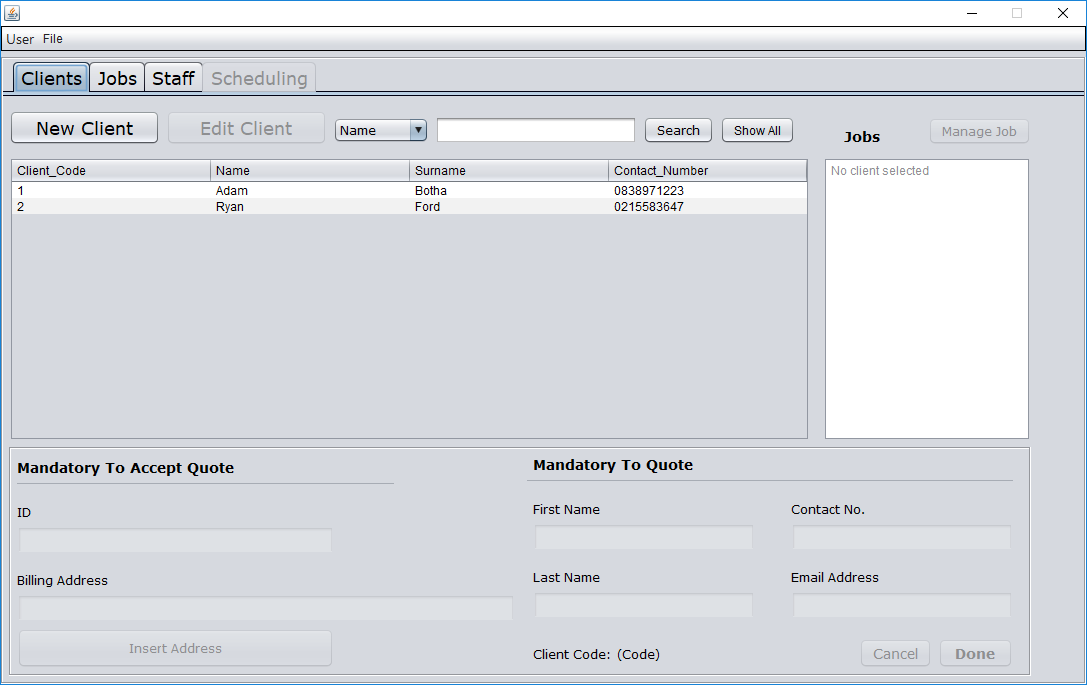
* + 1. Select Finish



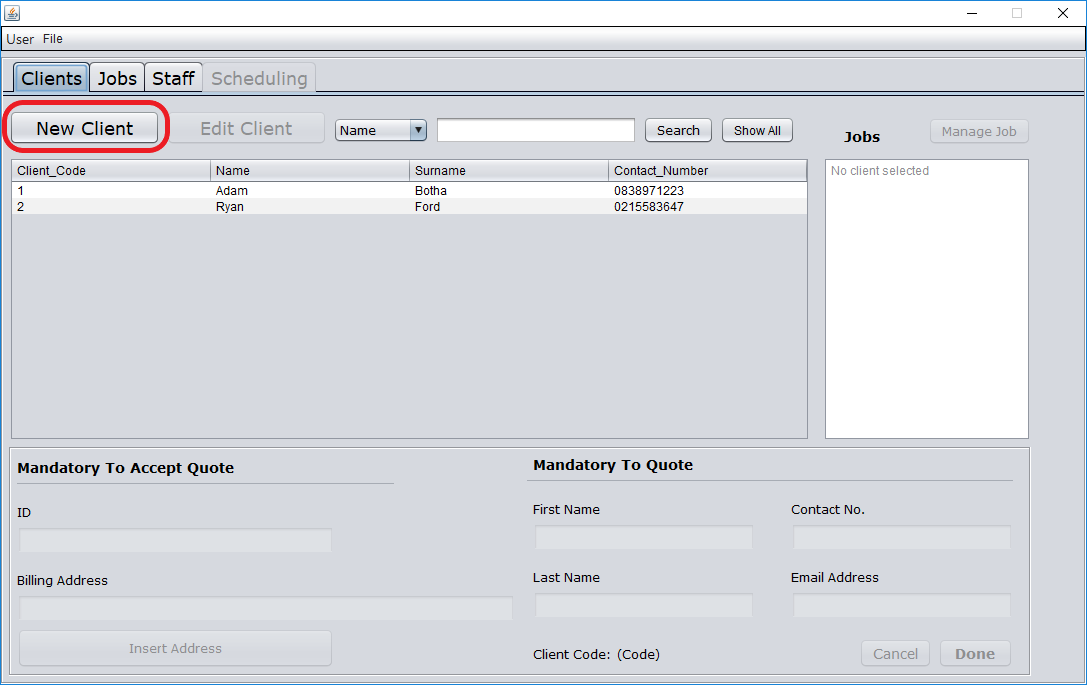
* + 1. Launch application



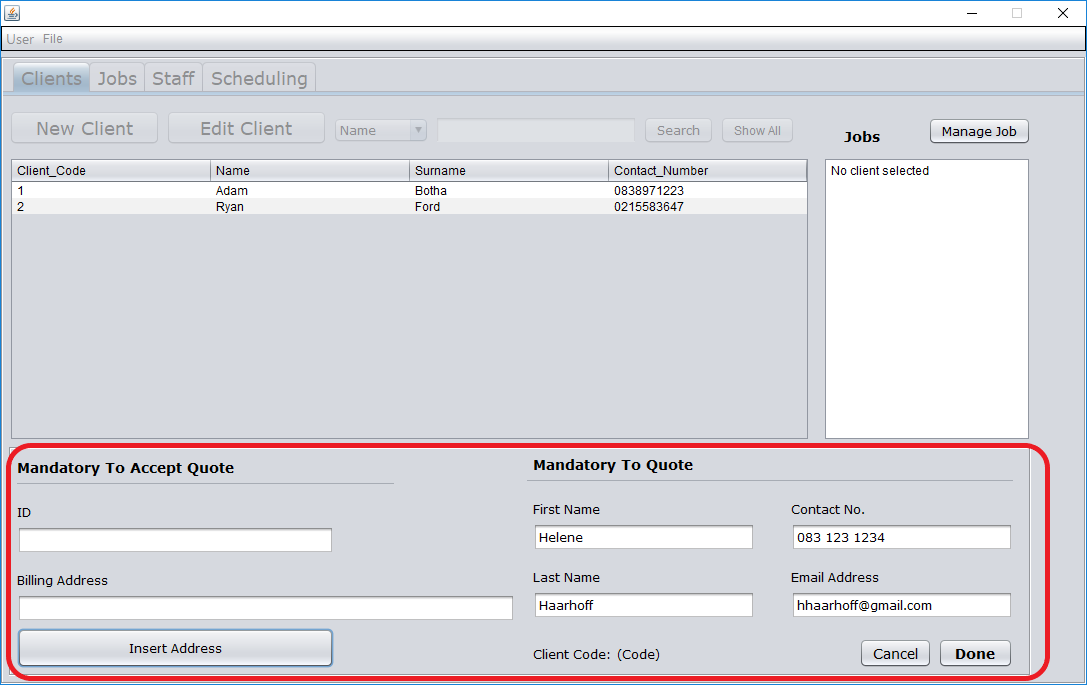
* 1. Instructions
     1. Add a new client
        1. Navigate to Clients Page



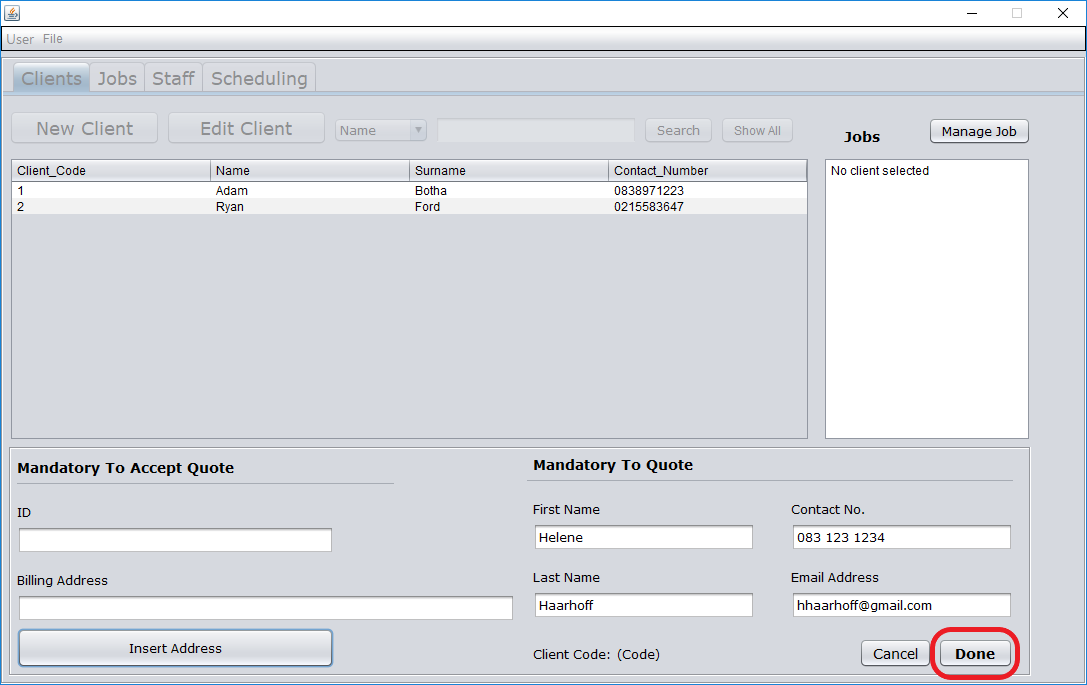
* + - 1. Select New Client



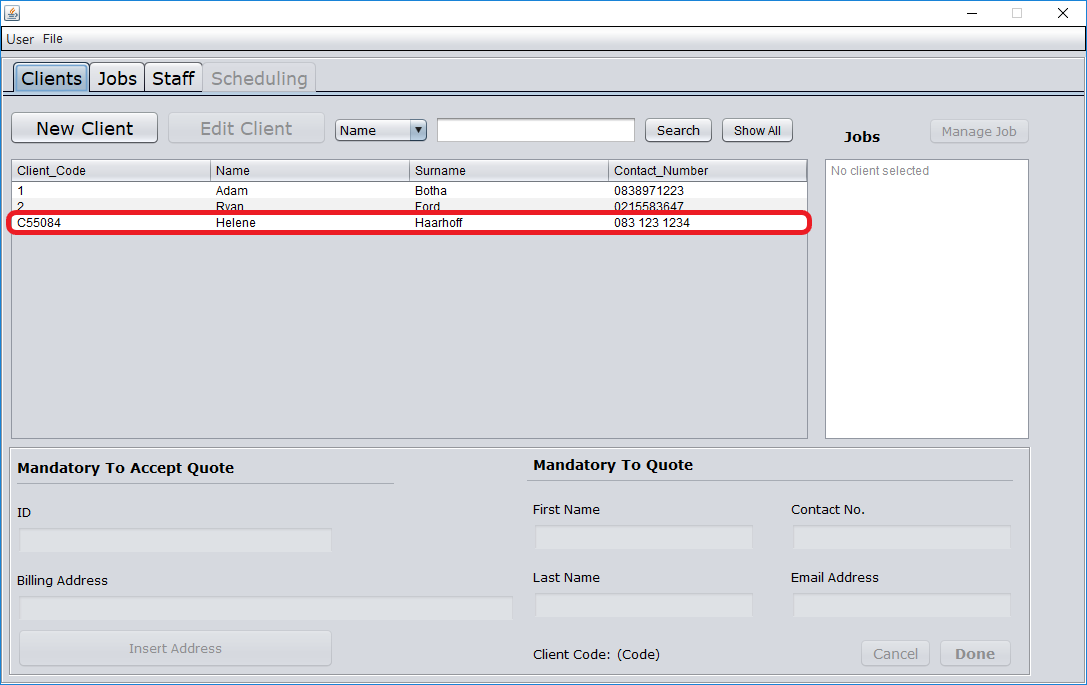
* + - 1. Enter available client details



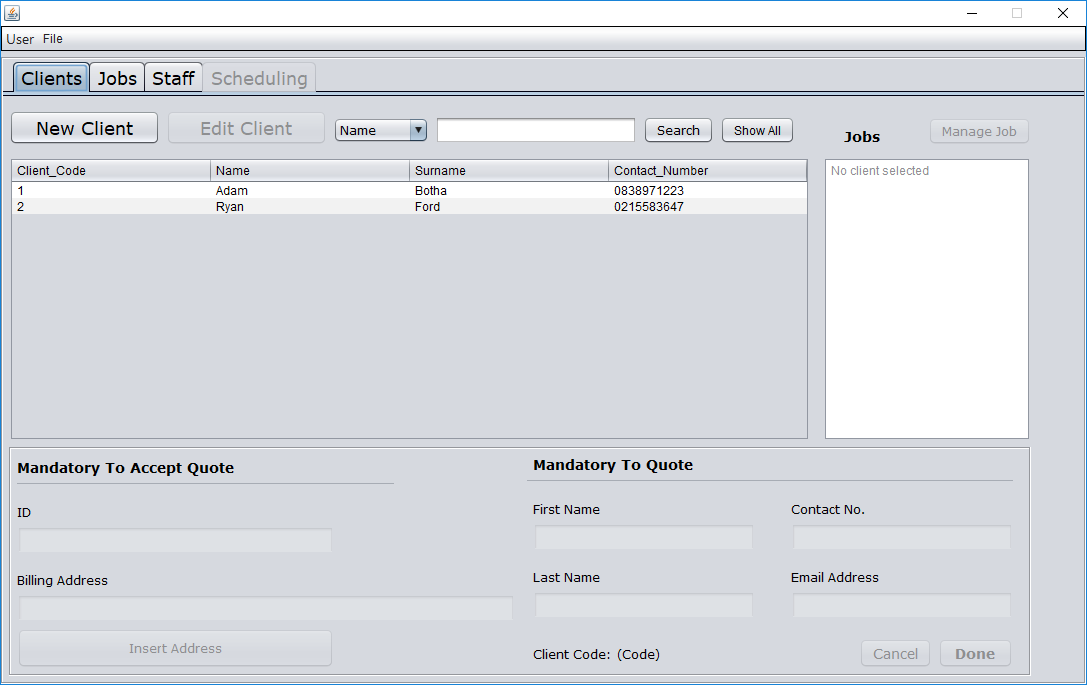
* + - 1. Select Done



* + - 1. Confirm that new client has been added



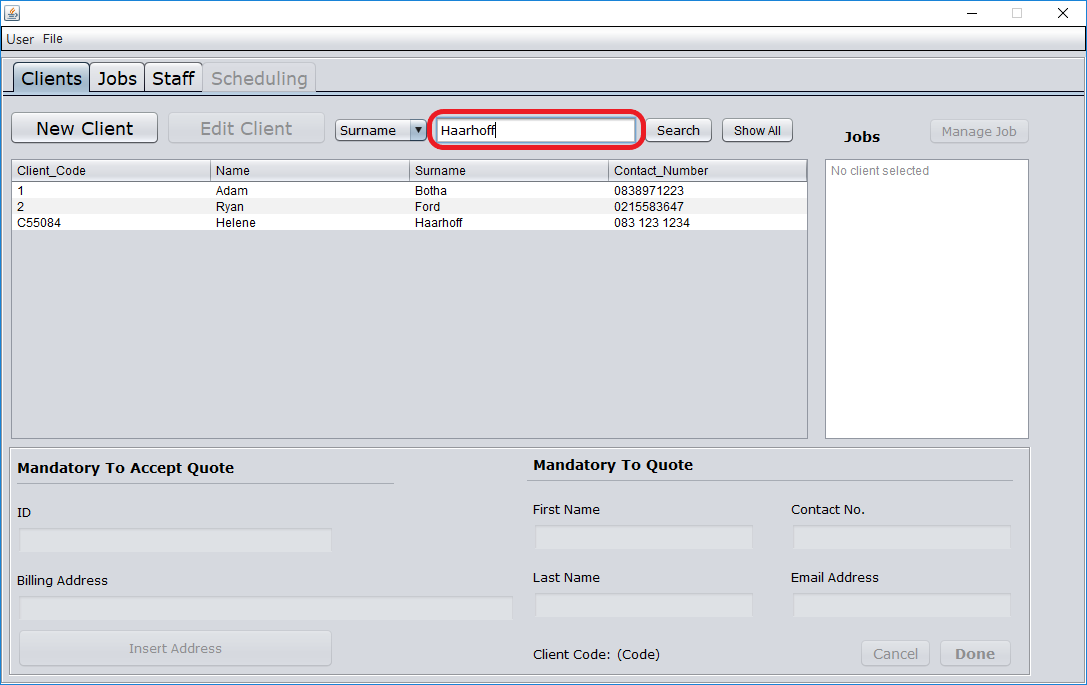
* + 1. Search for a client
       1. Navigate to Clients Page



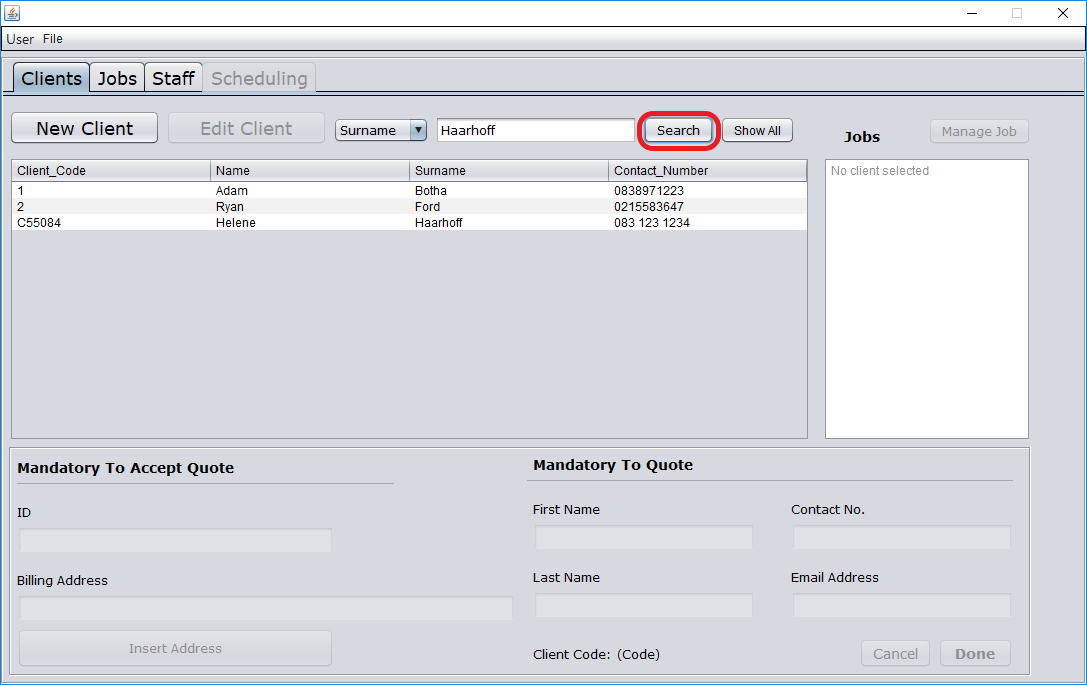
* + - 1. Select search metric



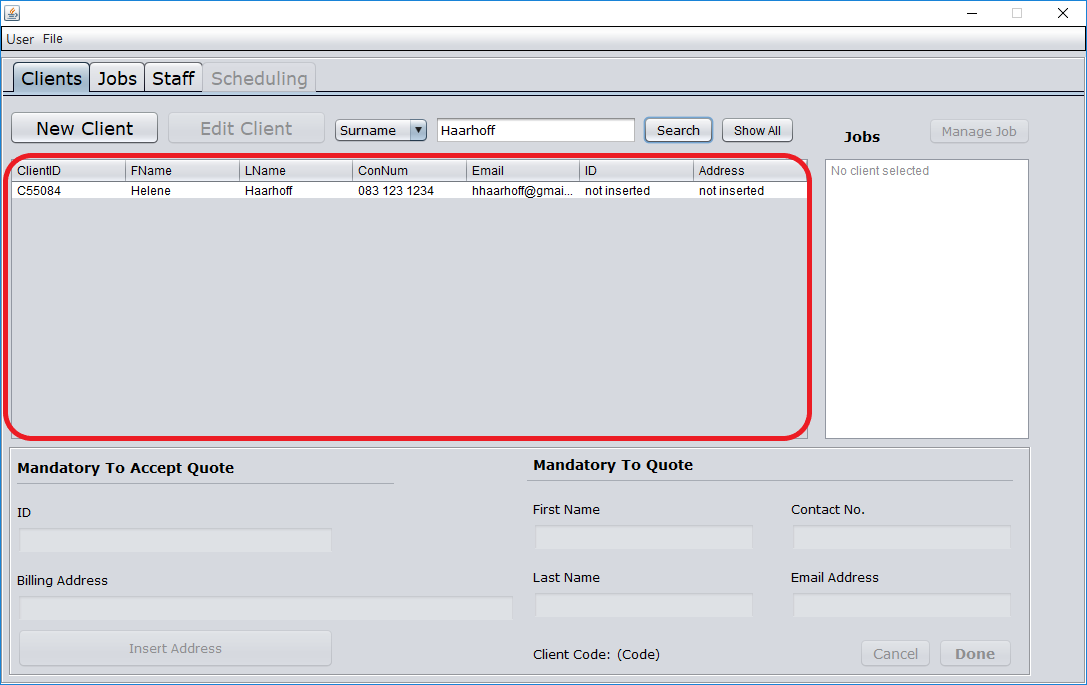
* + - 1. Enter search term



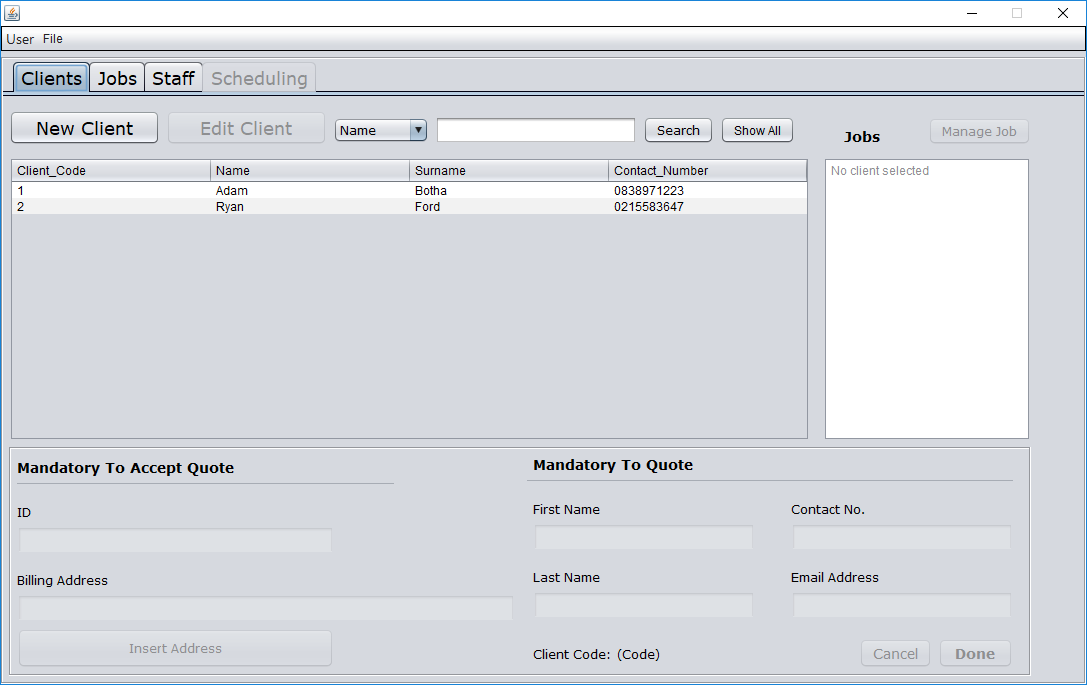
* + - 1. Select Search



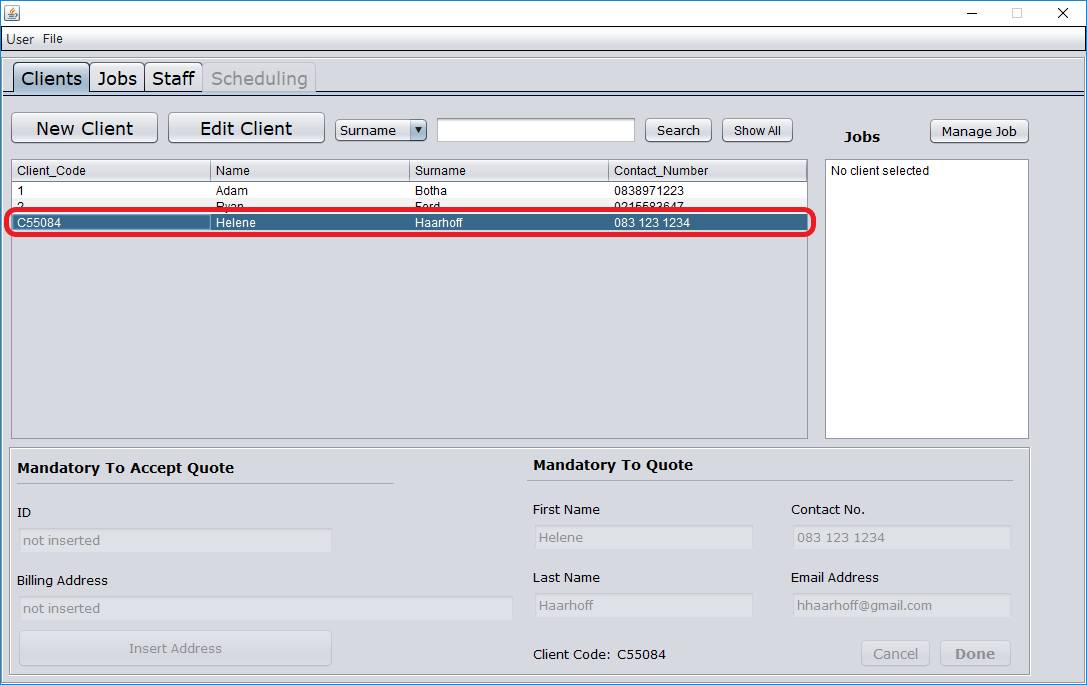
* + - 1. View search results



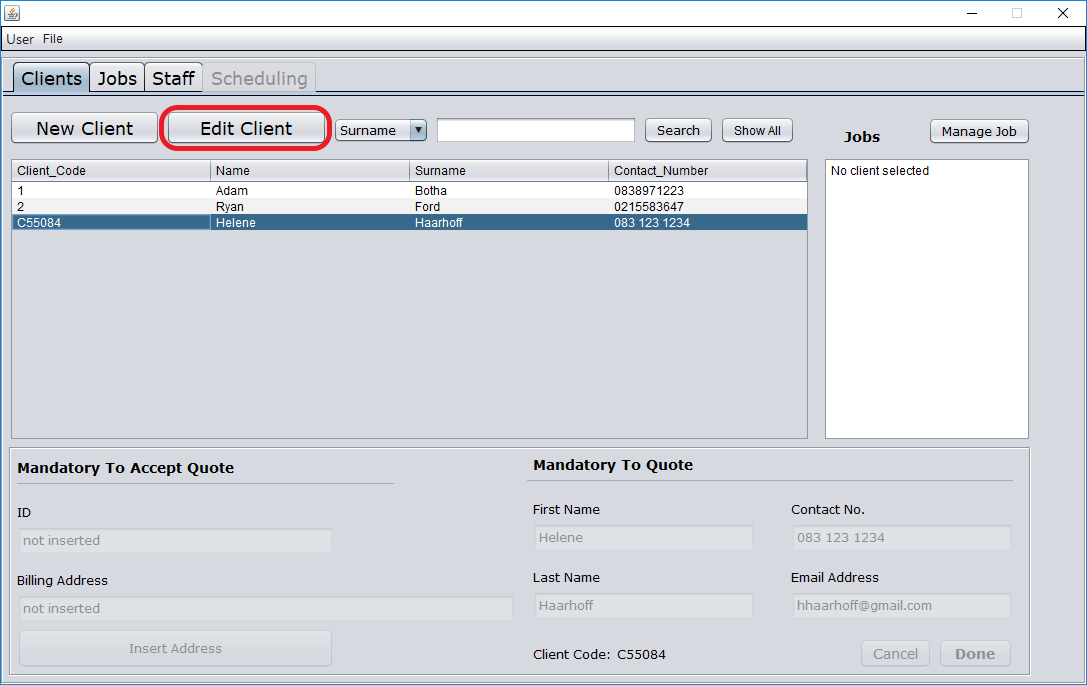
* + 1. Edit an existing client
       1. Navigate to Clients Page



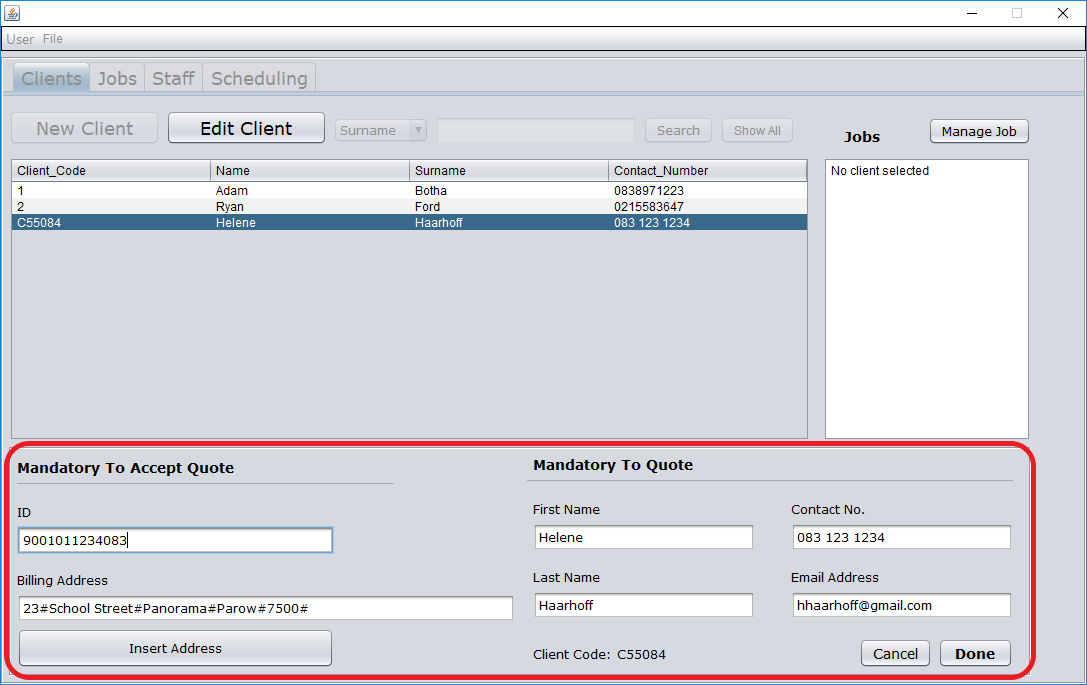
* + - 1. Select a client



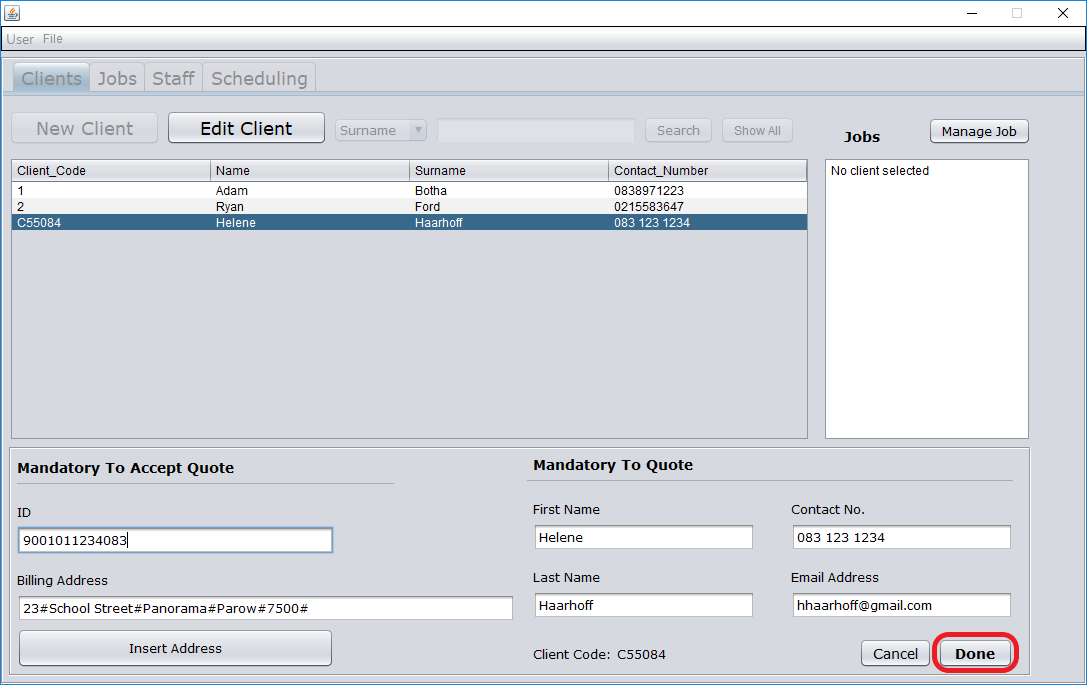
* + - 1. Select Edit Client



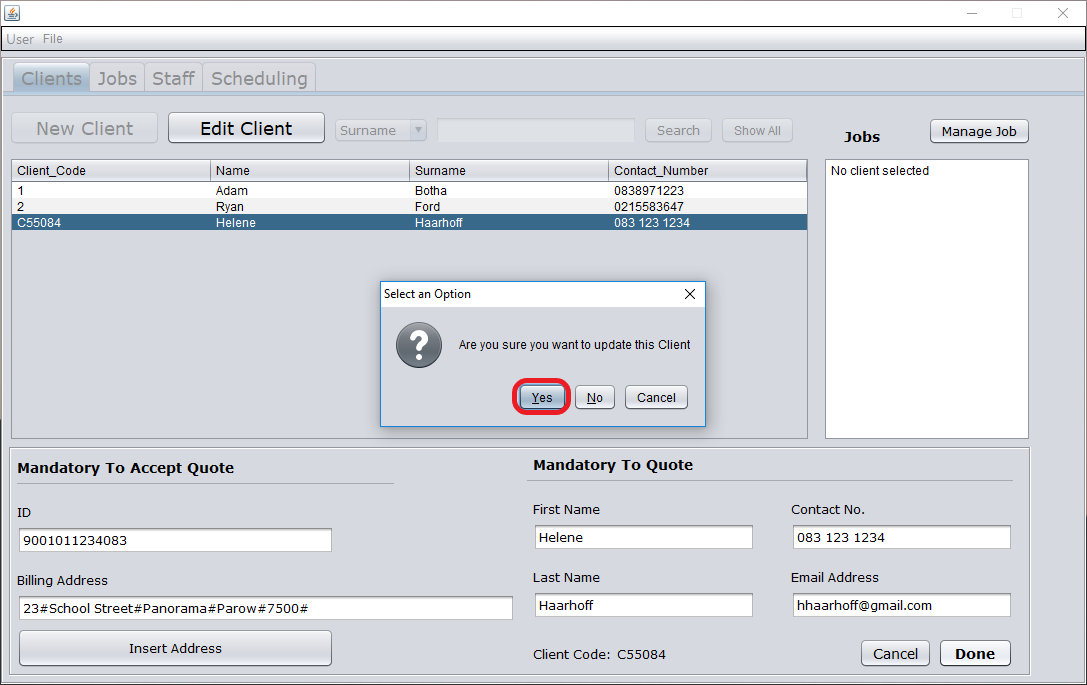
* + - 1. Update client details



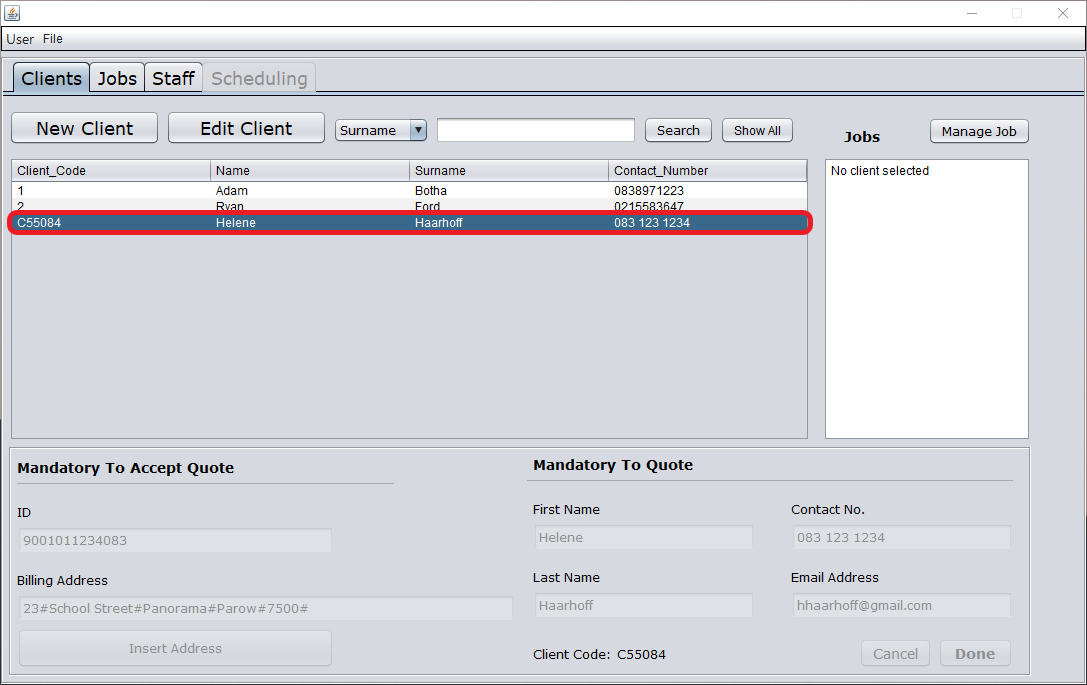
* + - 1. Select Done



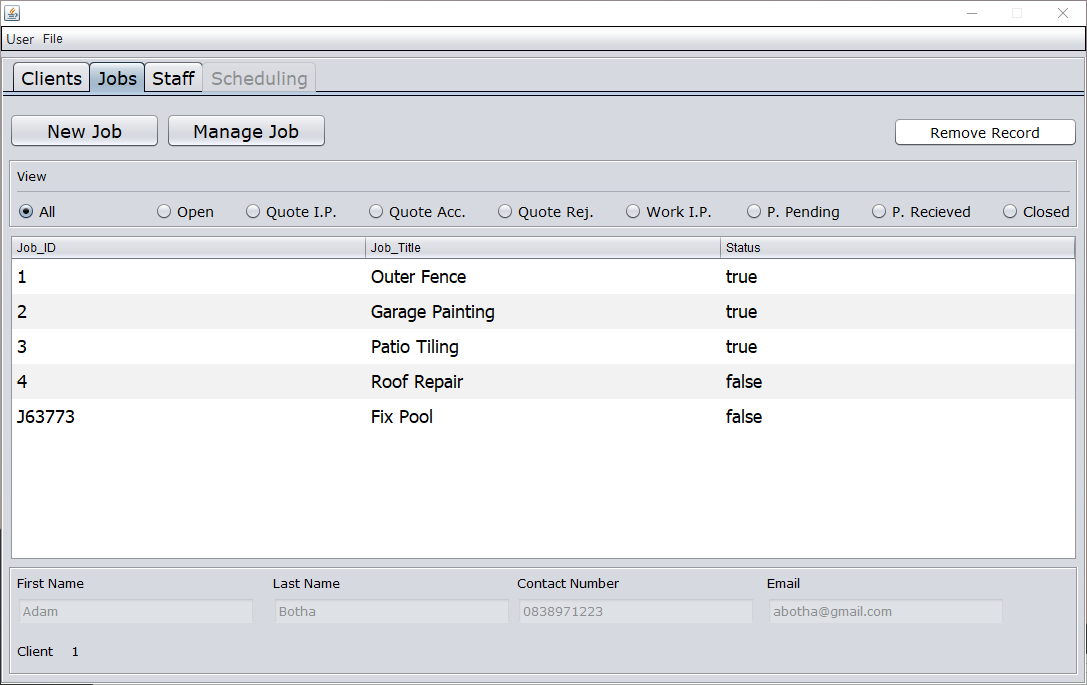
* + - 1. Select Yes



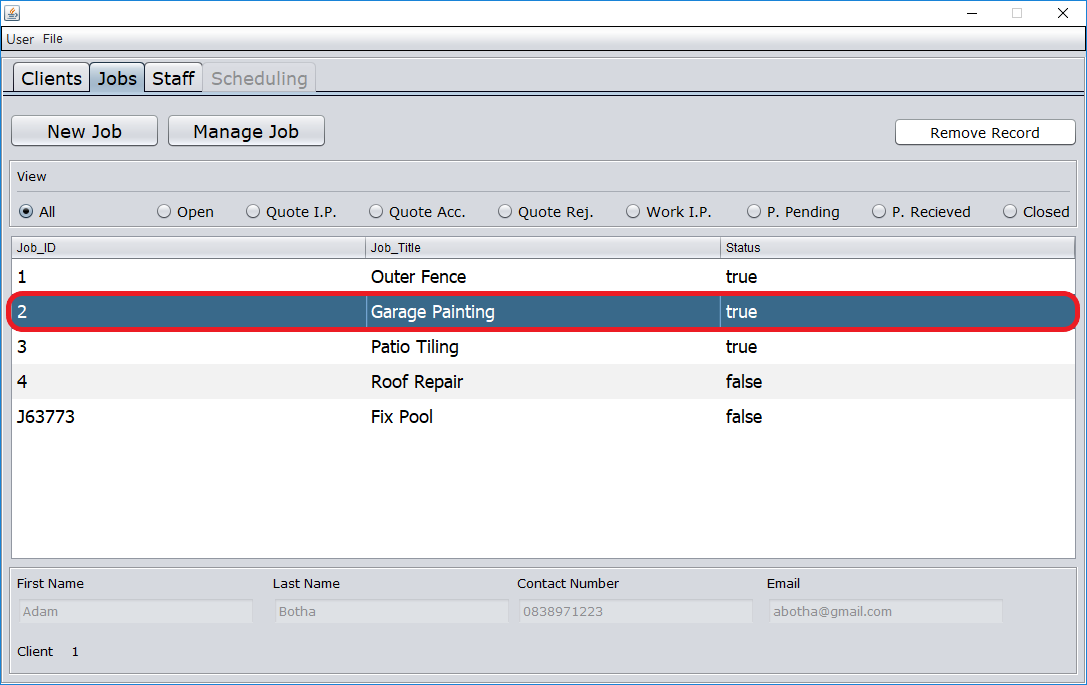
* + - 1. Select client to confirm updated details



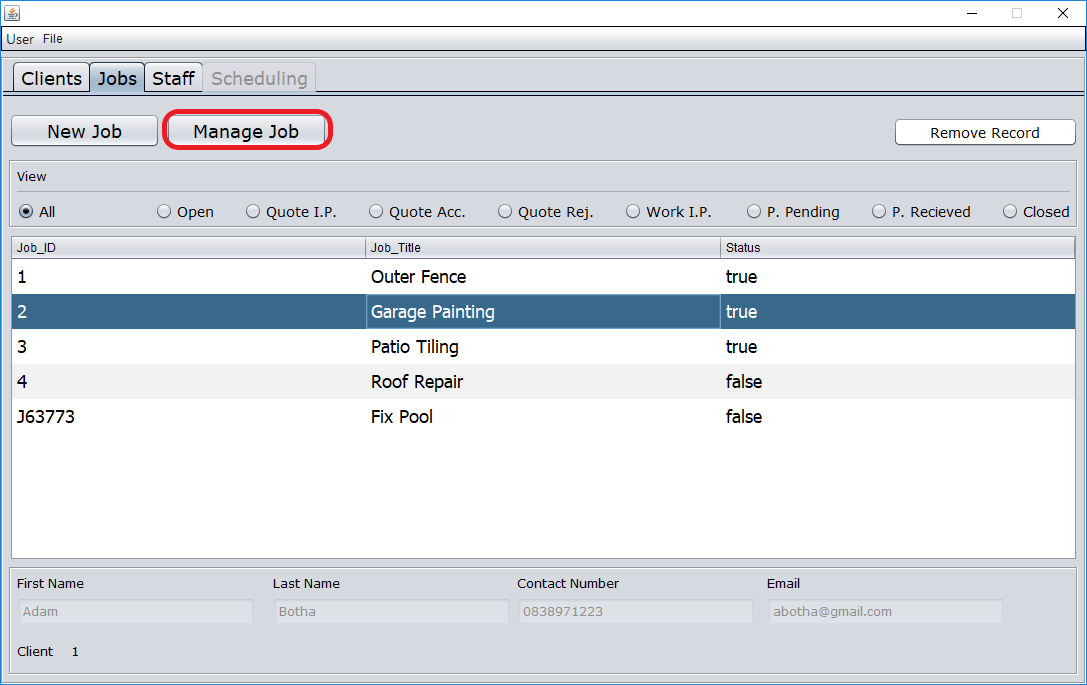
* + 1. Manage an existing job
       1. Navigate to Jobs Page



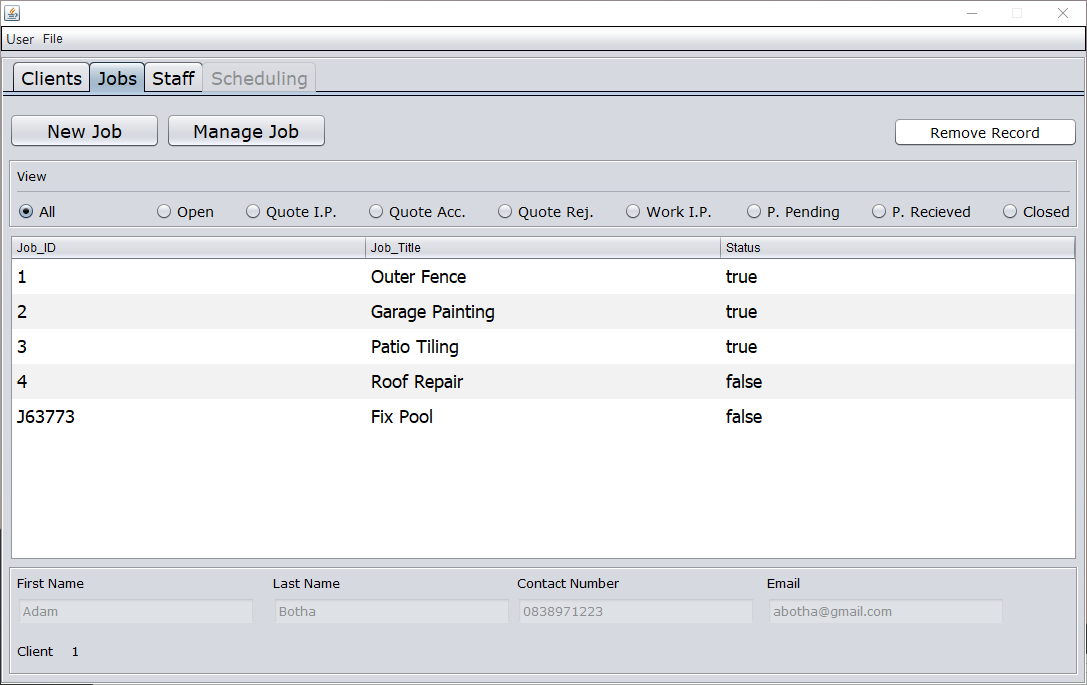
* + - 1. Select job to manage



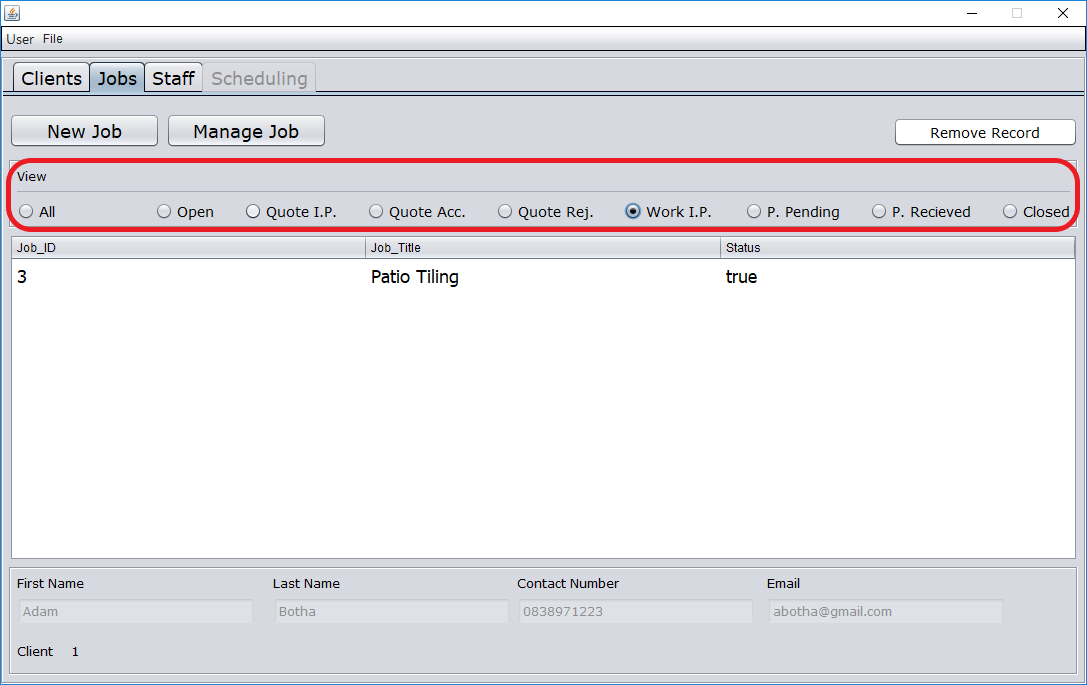
* + - 1. Select Manage Job



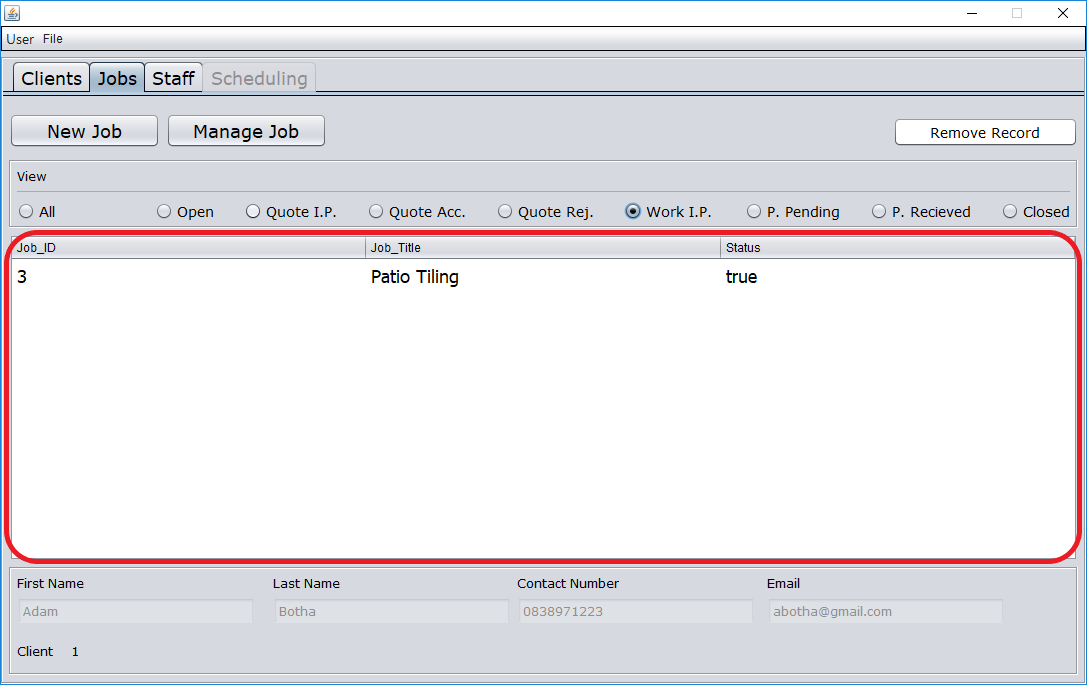
* + 1. View jobs per phase
       1. Navigate to Jobs Page



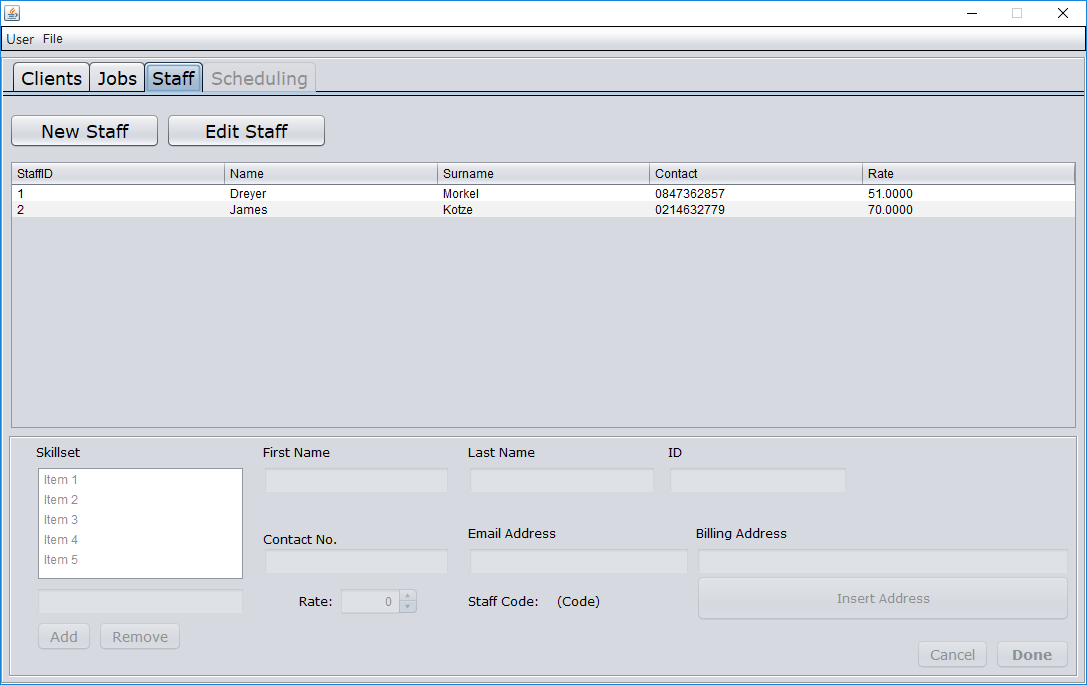
* + - 1. Select phase



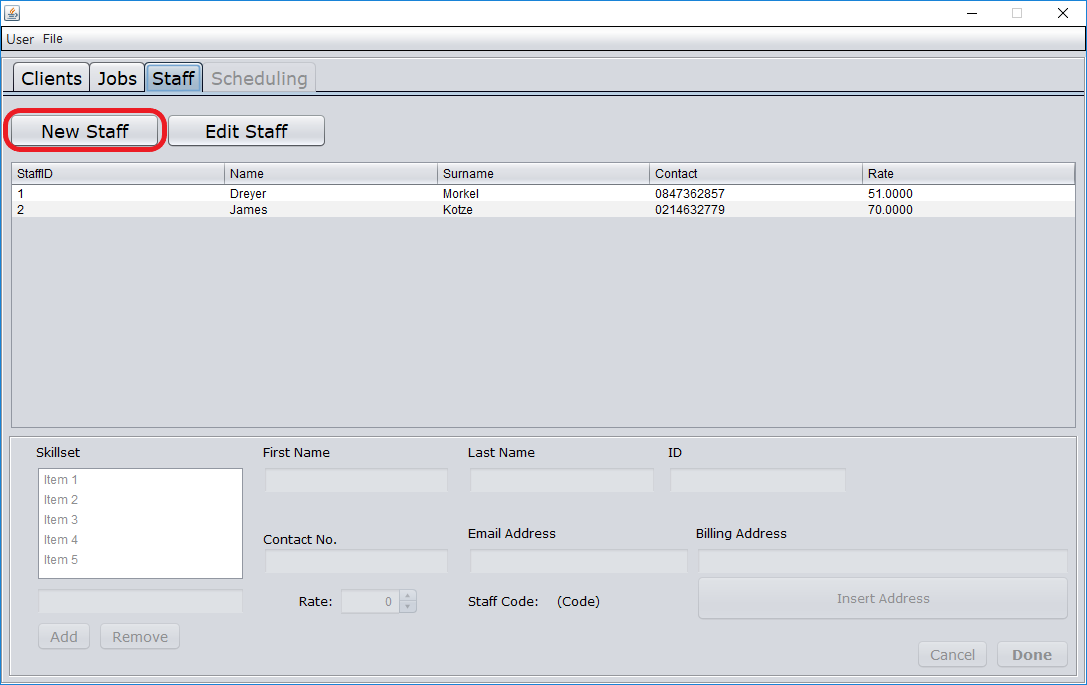
* + - 1. View jobs



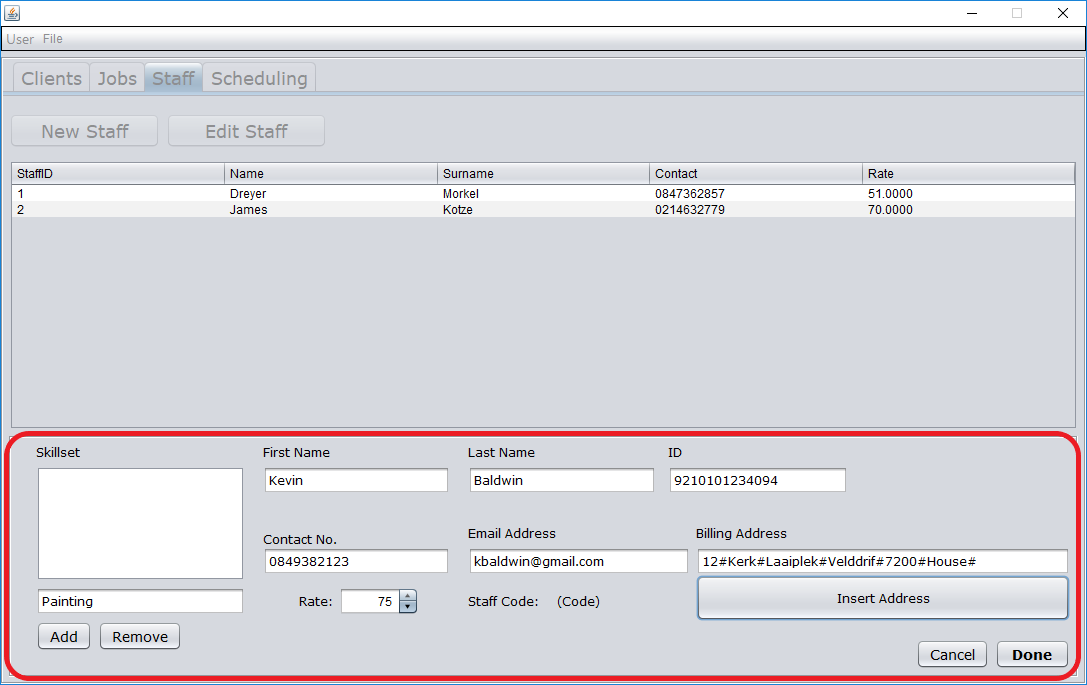
* + 1. Add a new staff member
       1. Navigate to the Staff Page



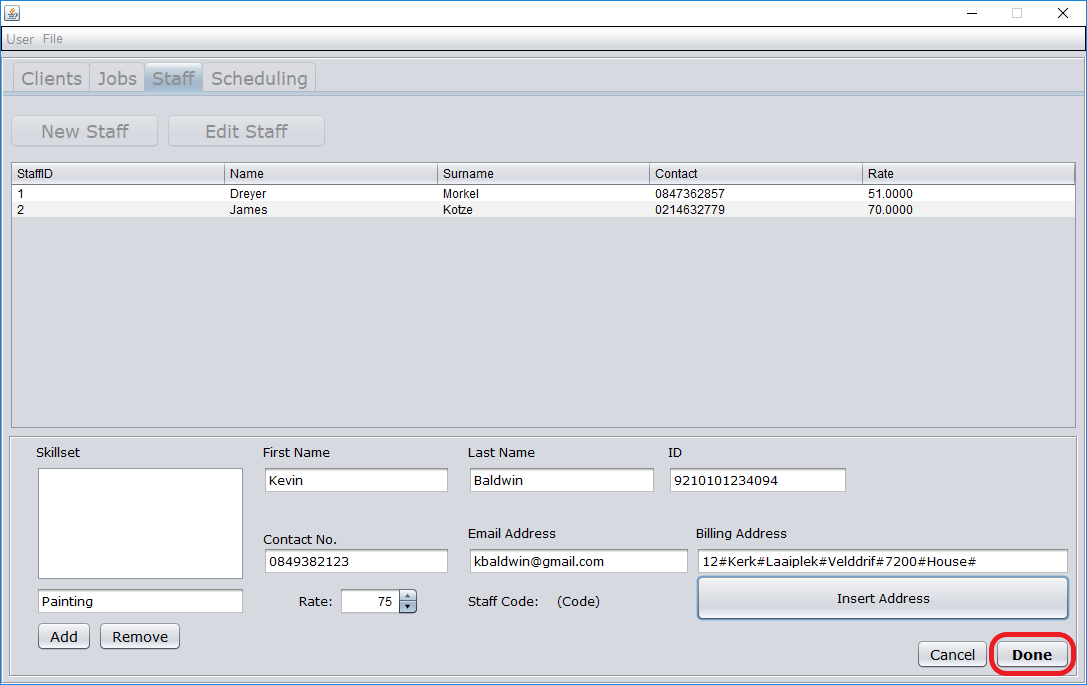
* + - 1. Select New Staff



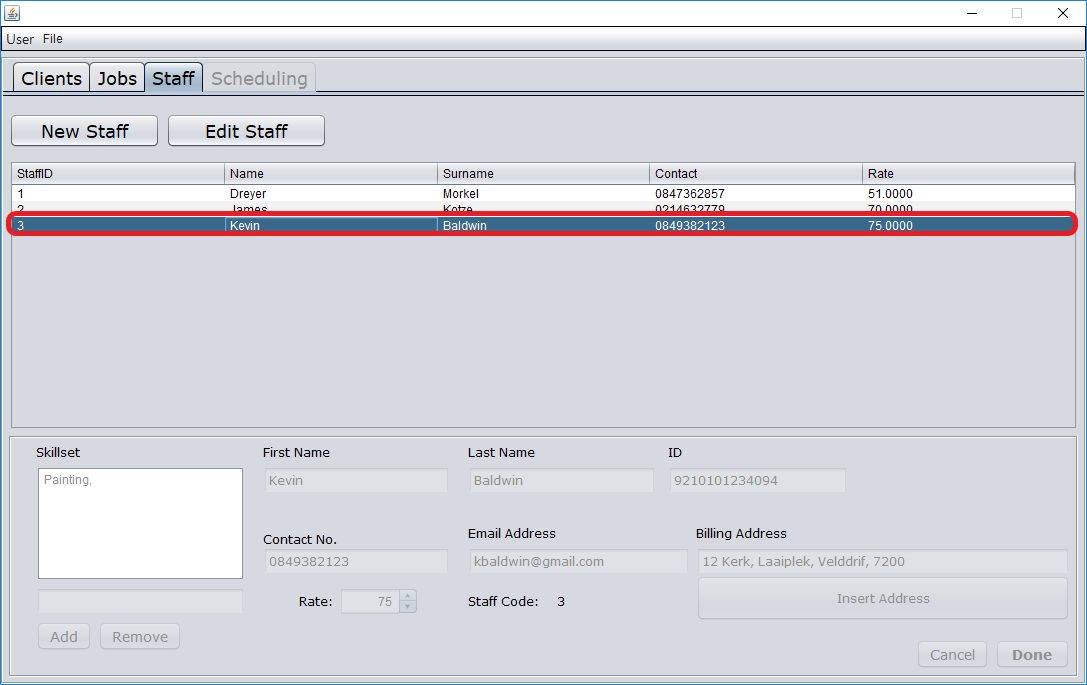
* + - 1. Enter staff details



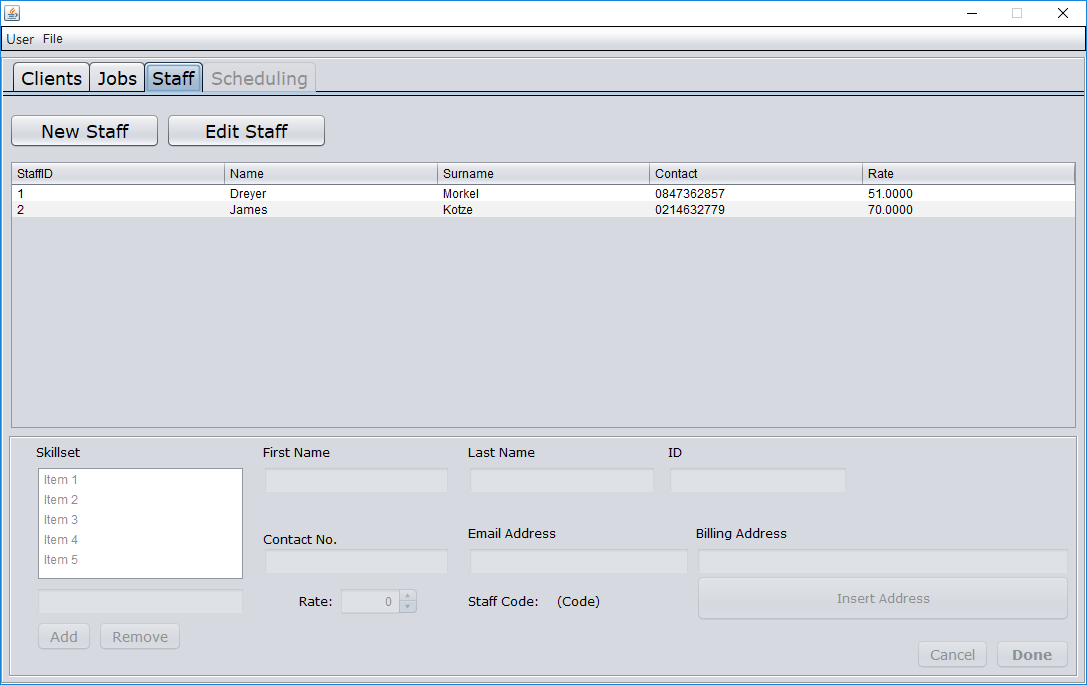
* + - 1. Select Done



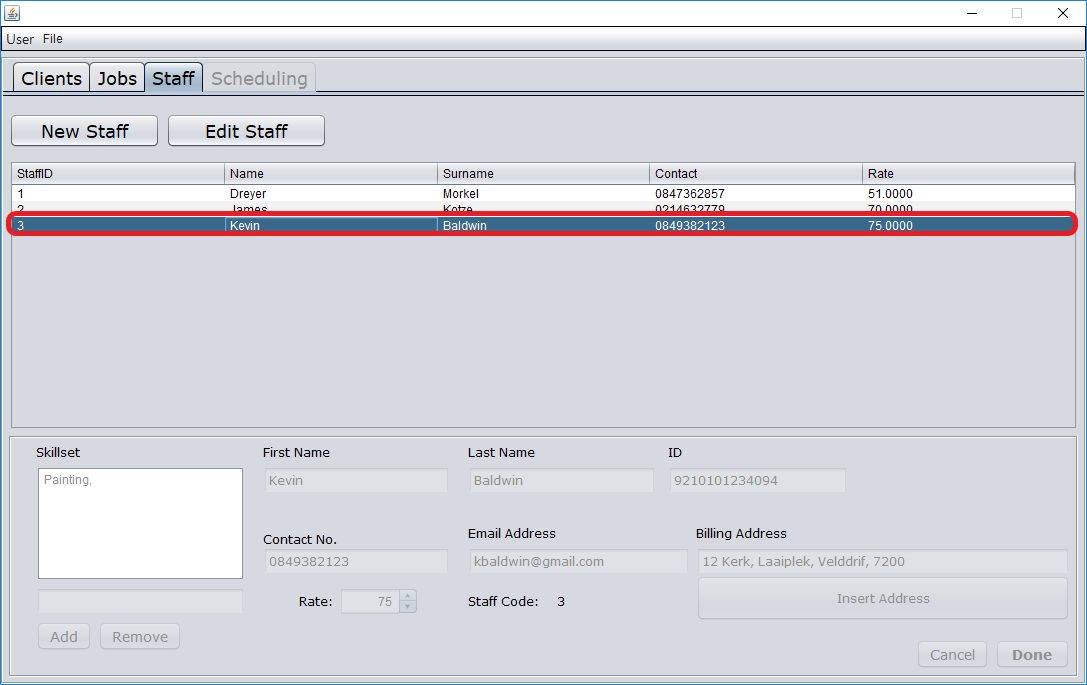
* + - 1. Confirm that new staff member has been added



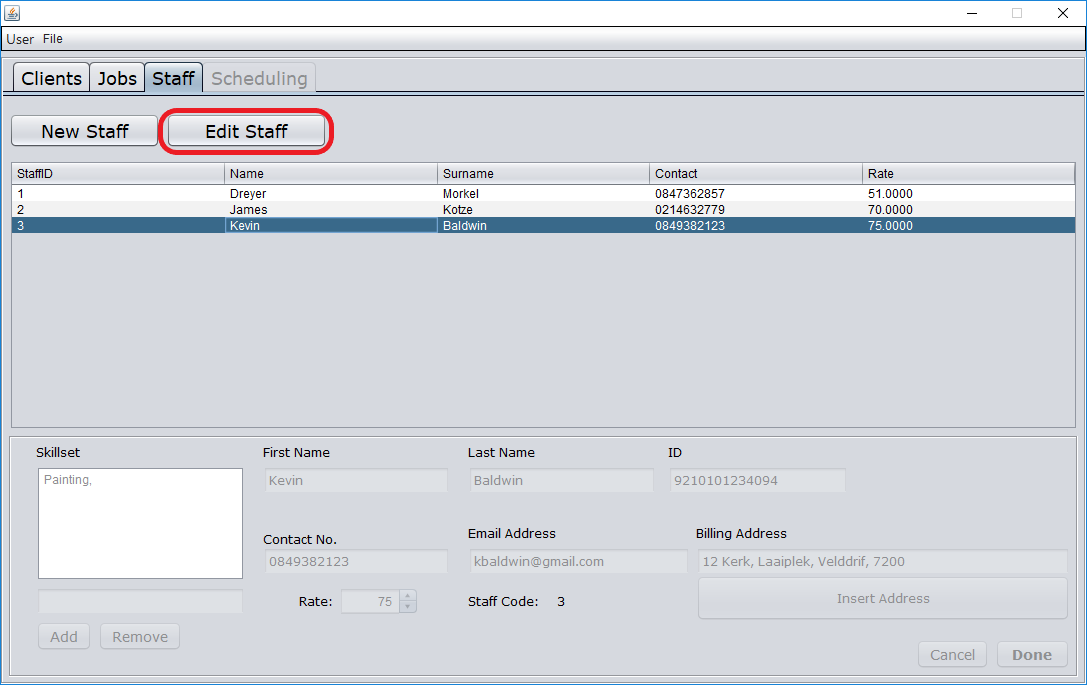
* + 1. Edit an existing staff member
       1. Navigate to the Staff Page



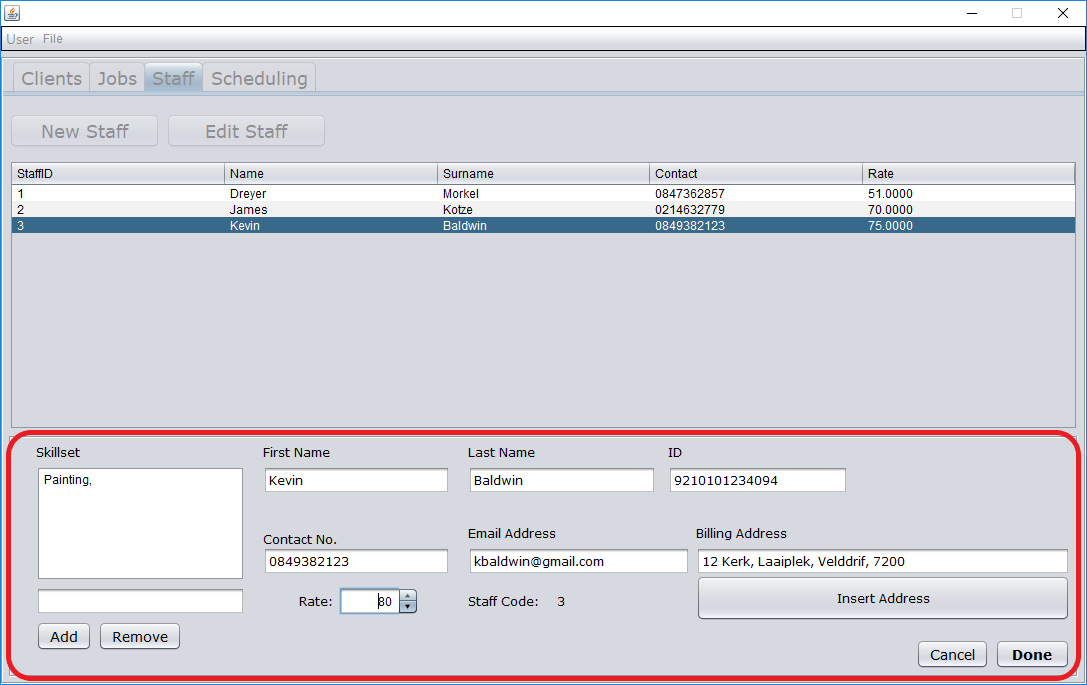
* + - 1. Select a staff member



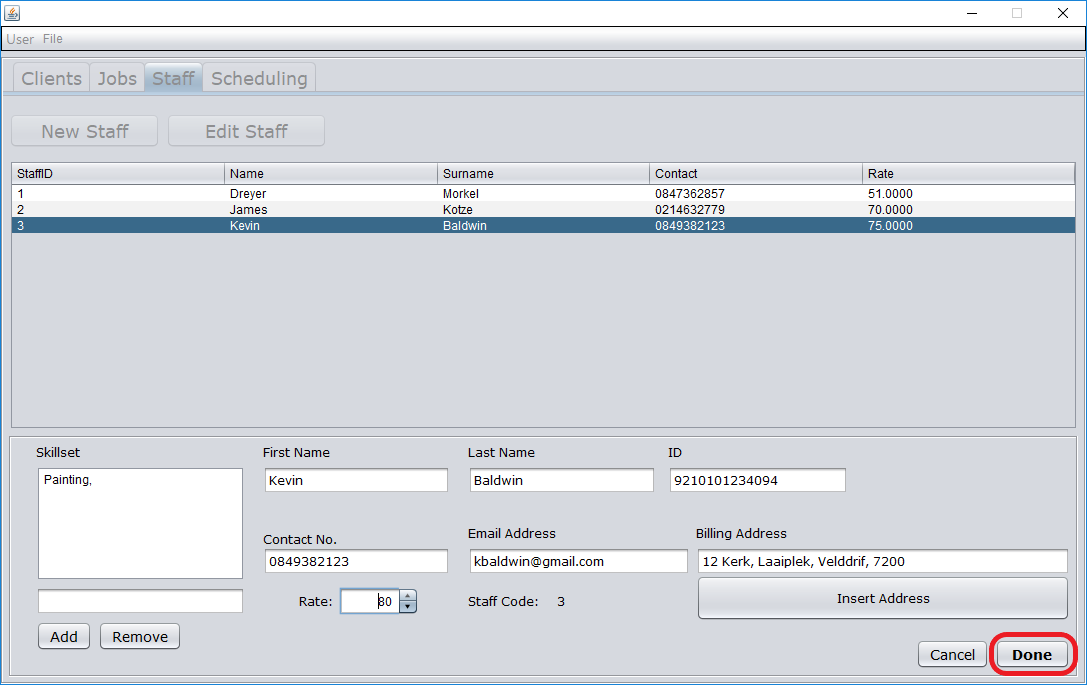
* + - 1. Select Edit Staff



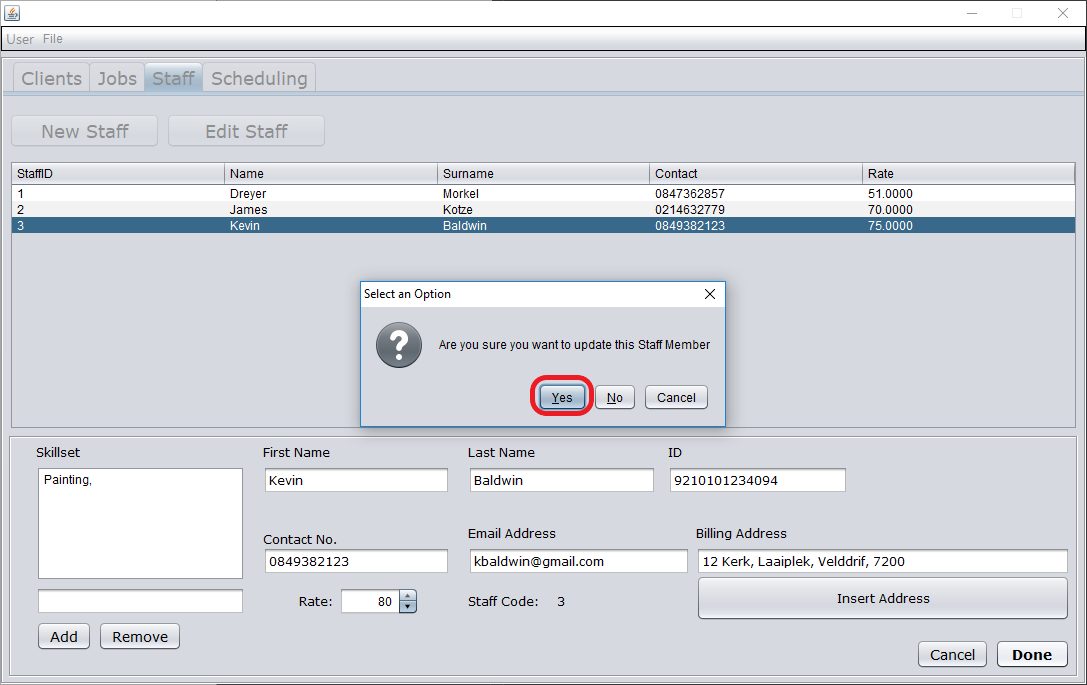
* + - 1. Update staff details



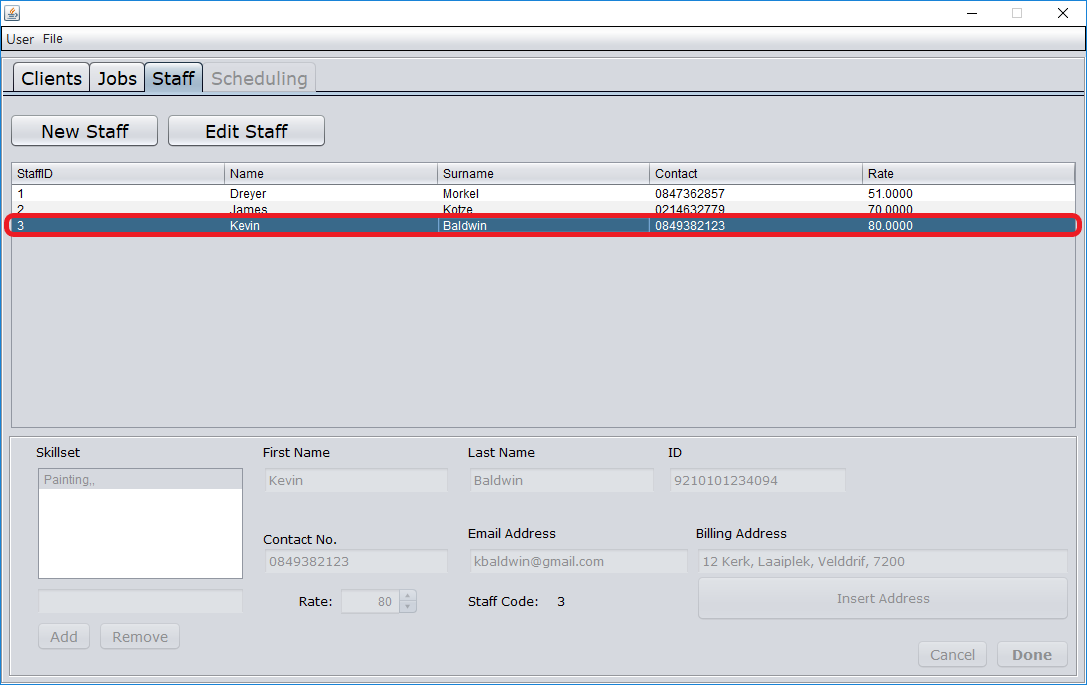
* + - 1. Select Done



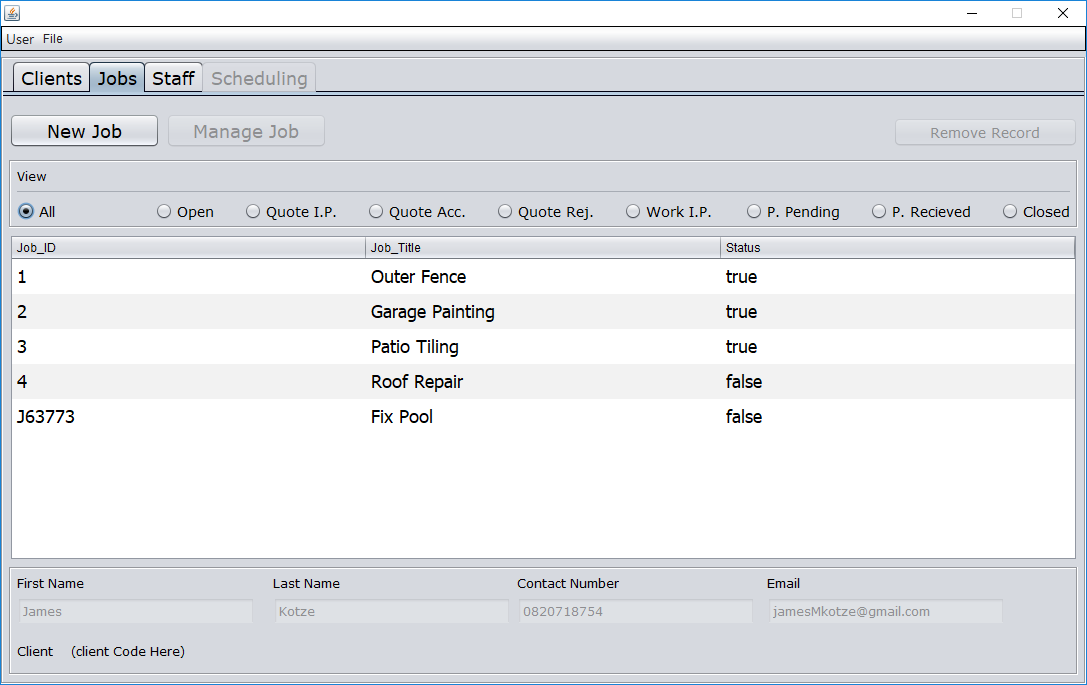
* + - 1. Select Yes



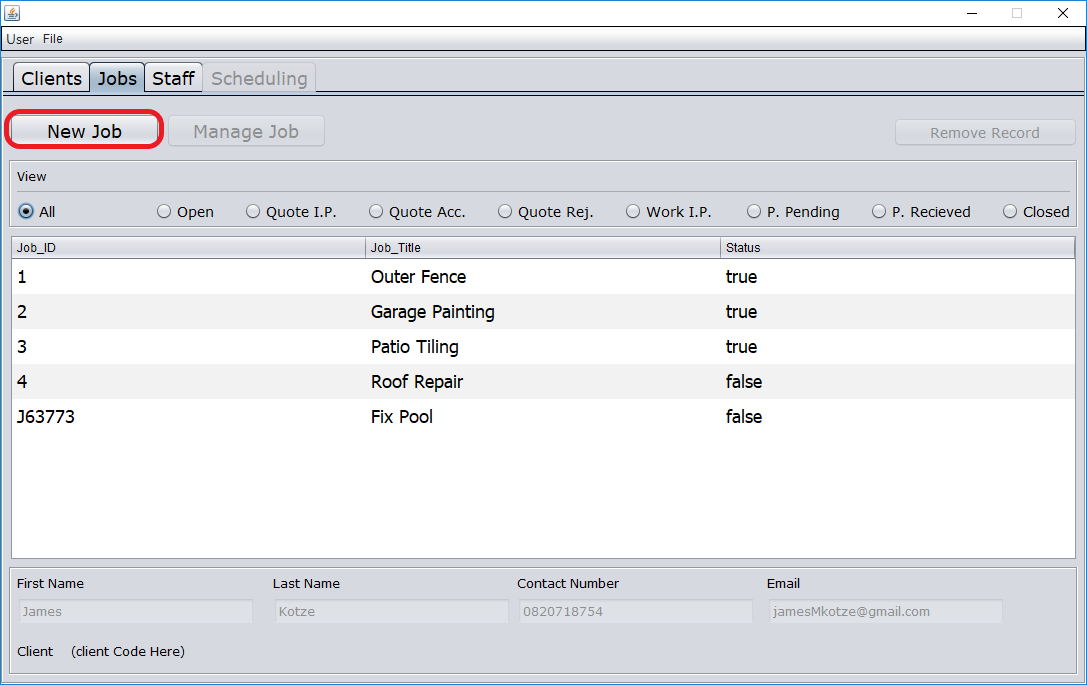
* + - 1. Select staff member to confirm updated details



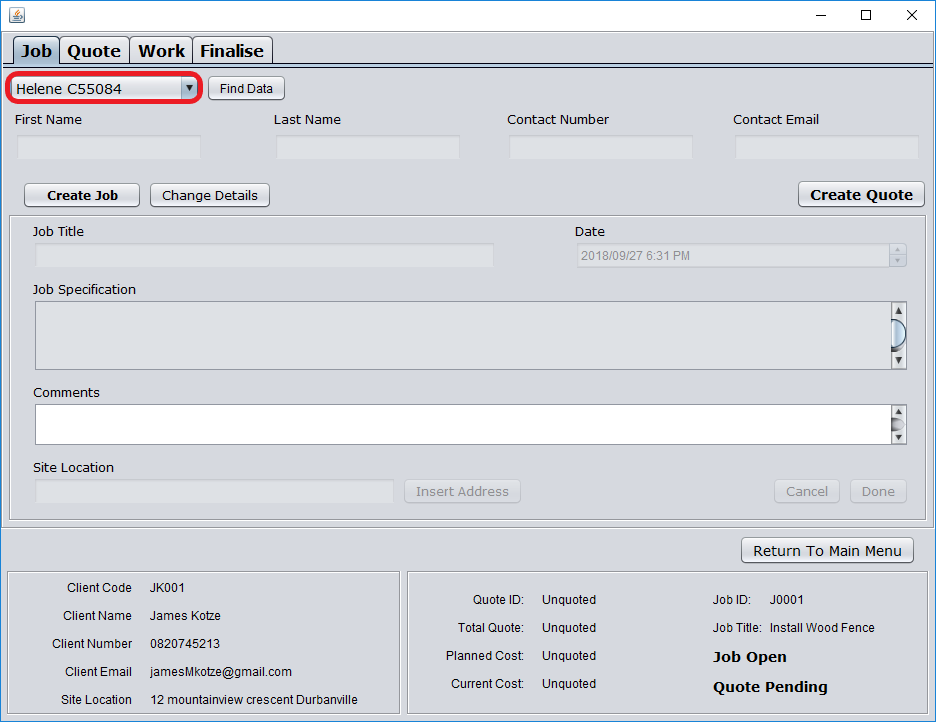
* + 1. Add a new job
       1. Navigate to Jobs Page



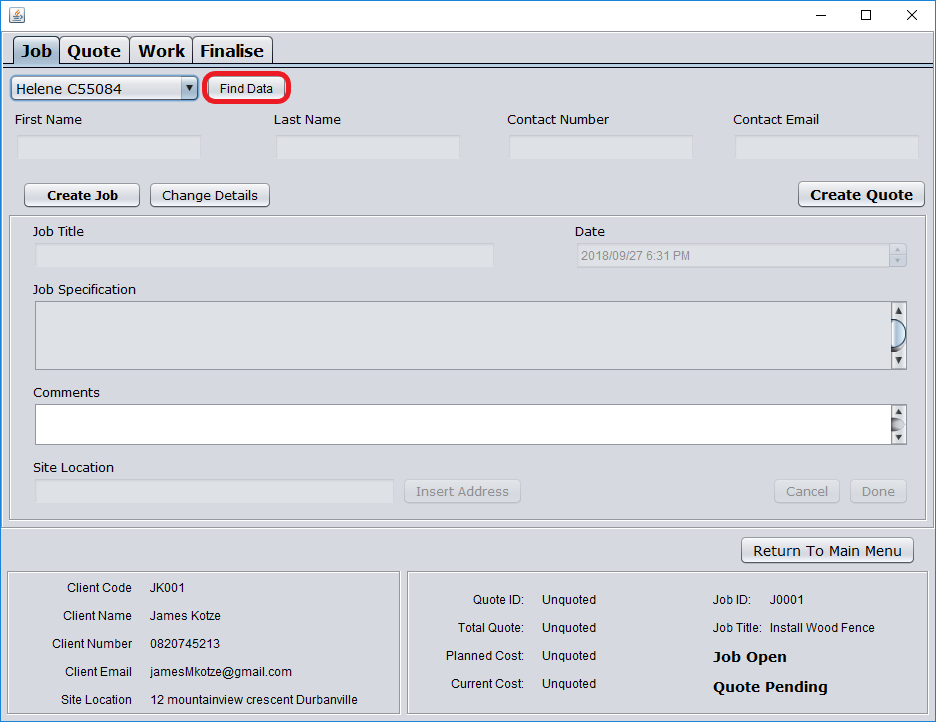
* + - 1. Select New Job



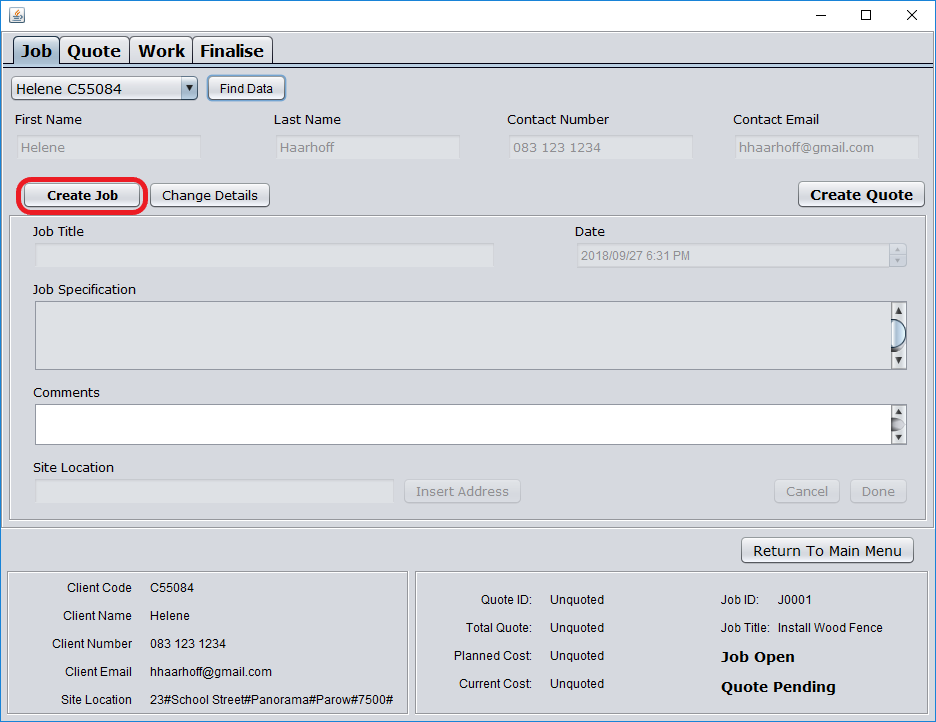
* + - 1. Select client



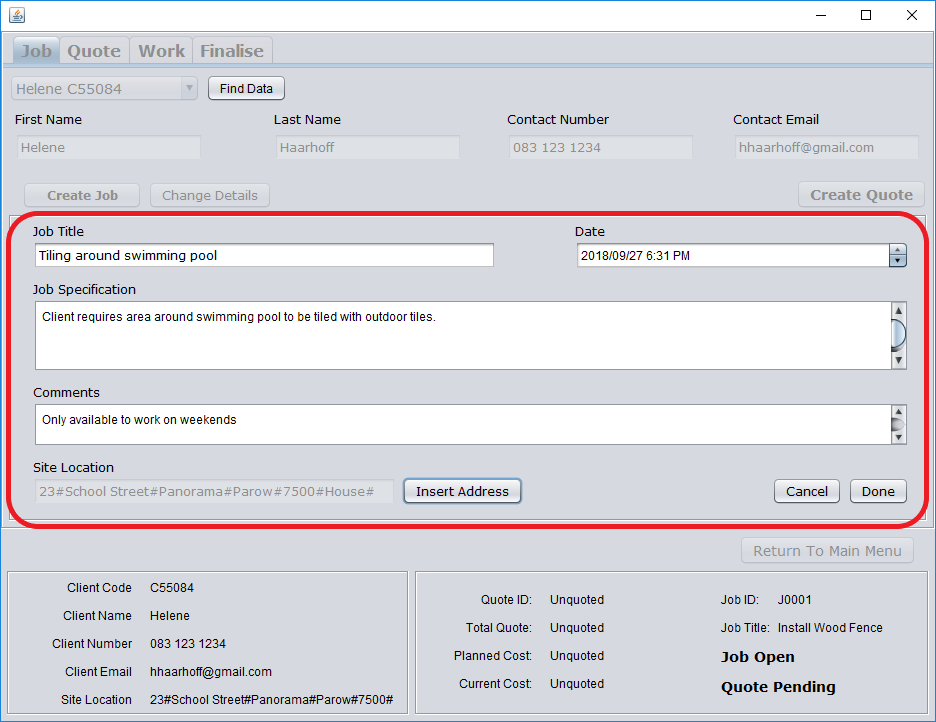
* + - 1. Select Find Data



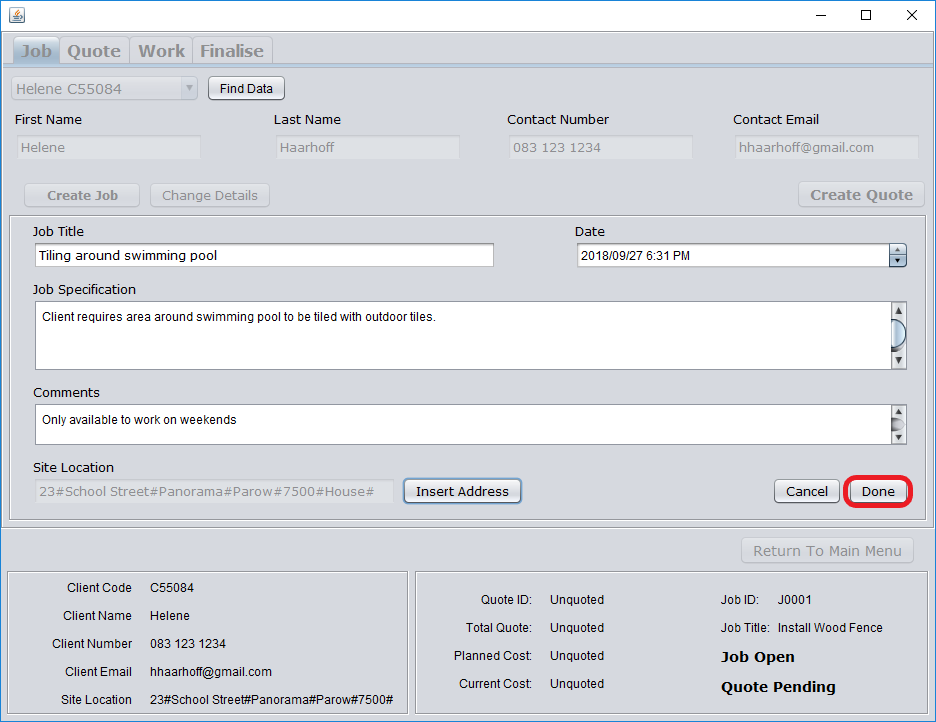
* + - 1. Select Create Job



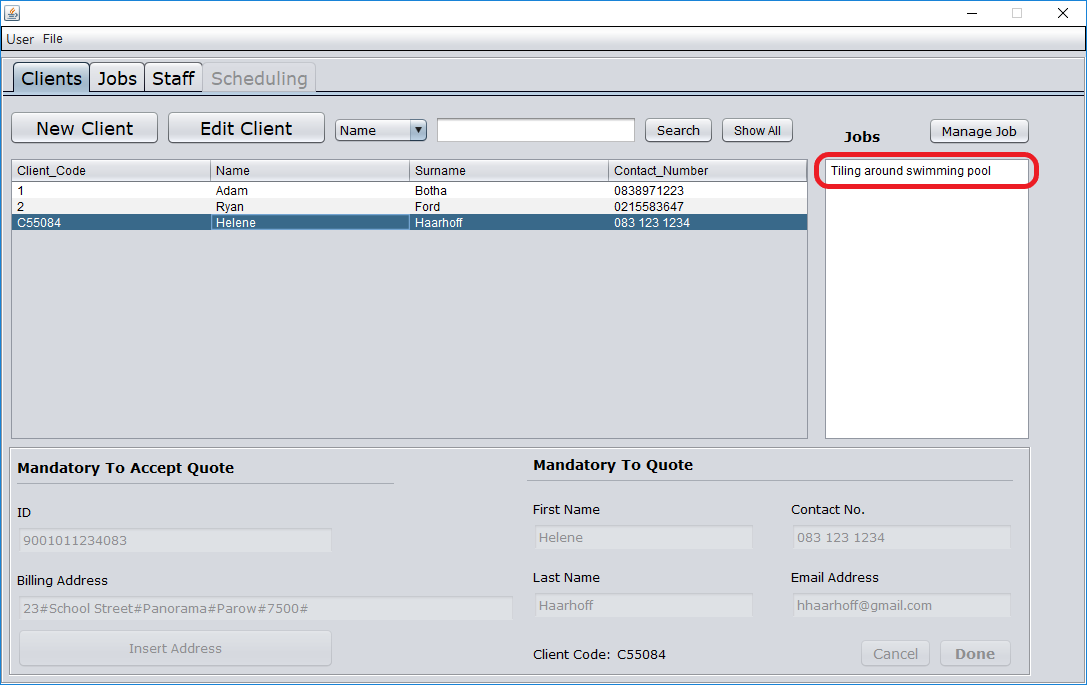
* + - 1. Enter job details



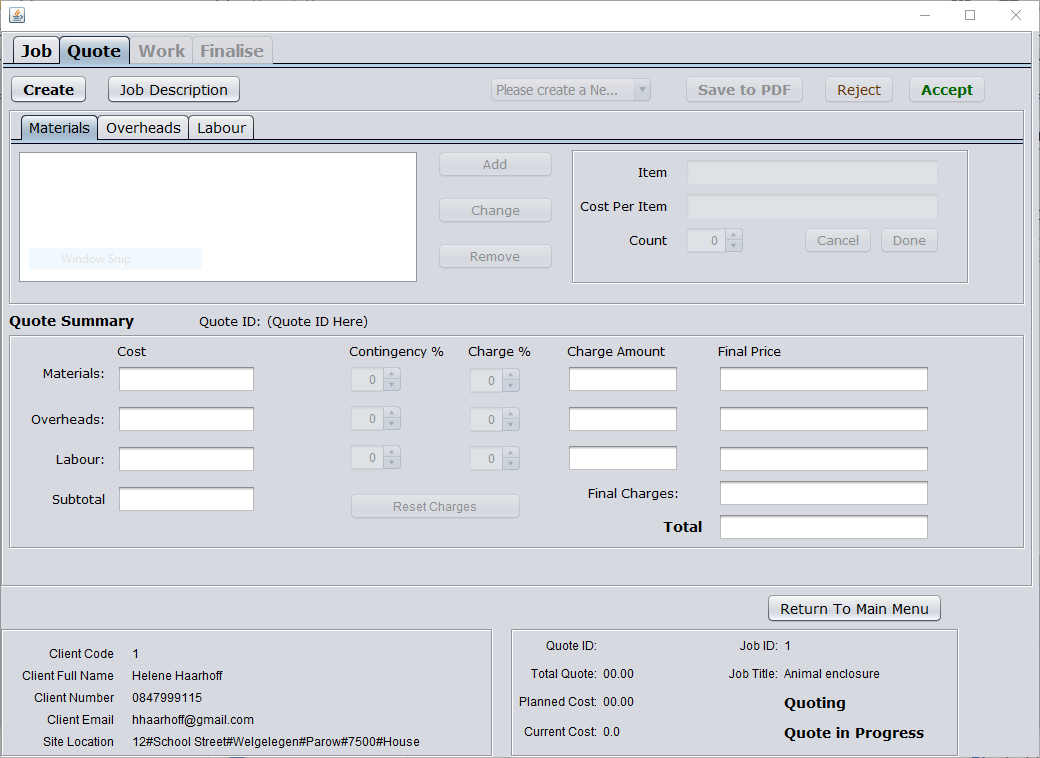
* + - 1. Select Done



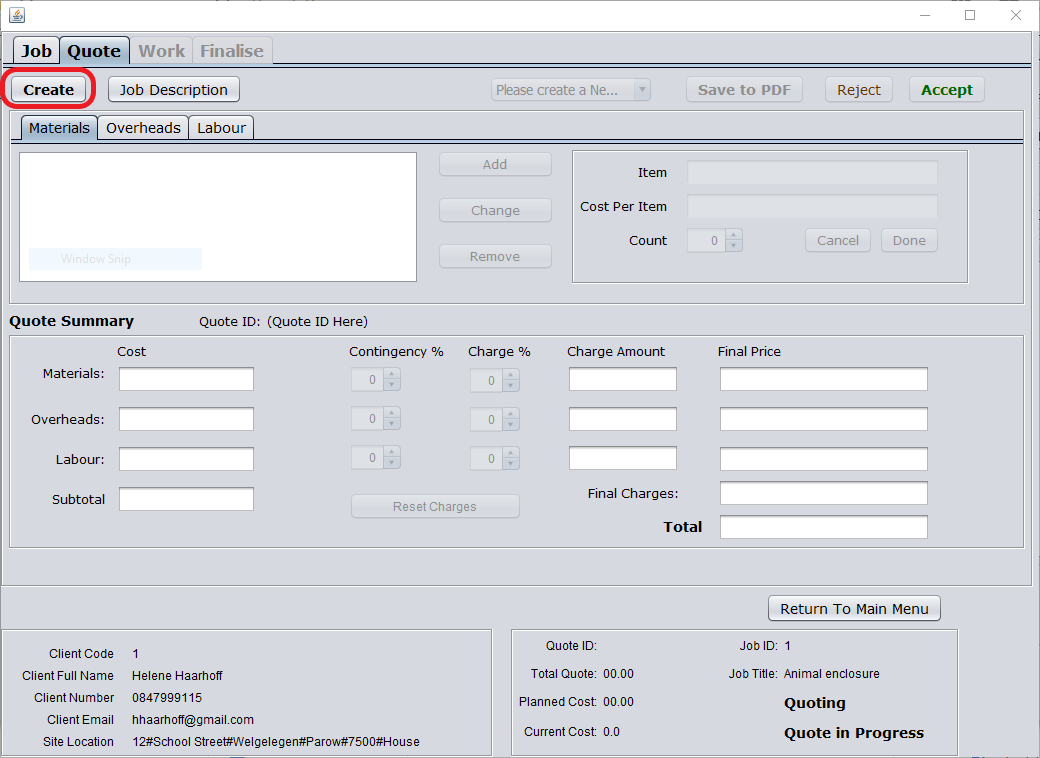
* + - 1. Confirm job has been added



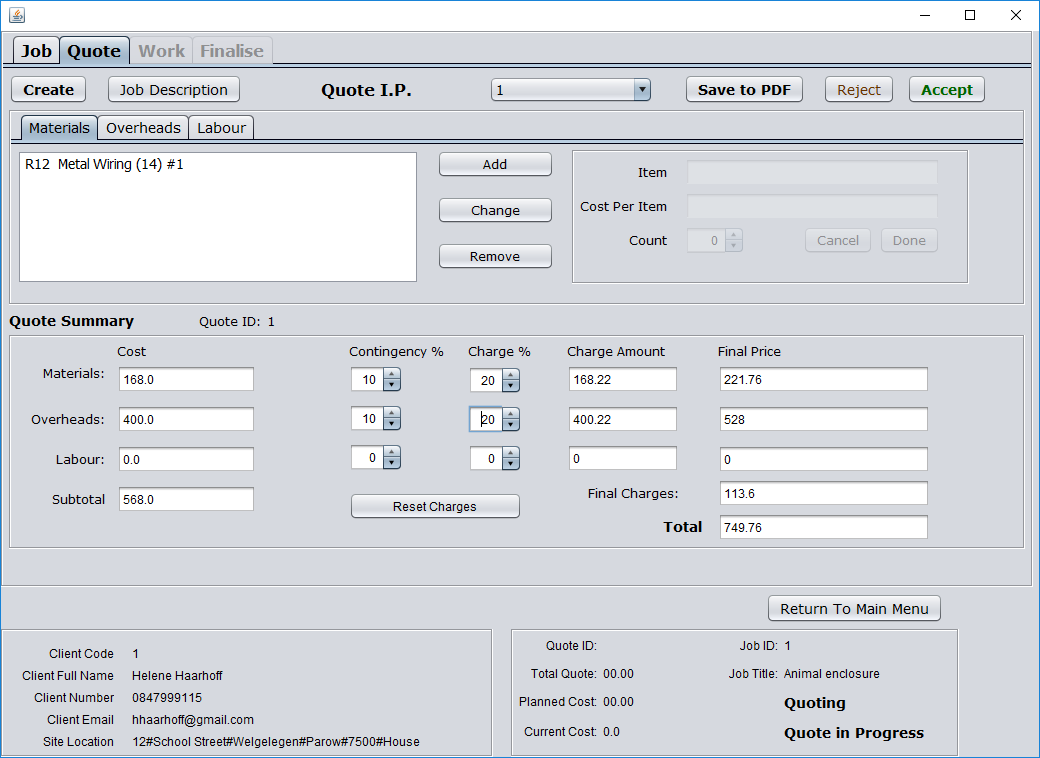
* + 1. Create a new quote
       1. Navigate to quote page



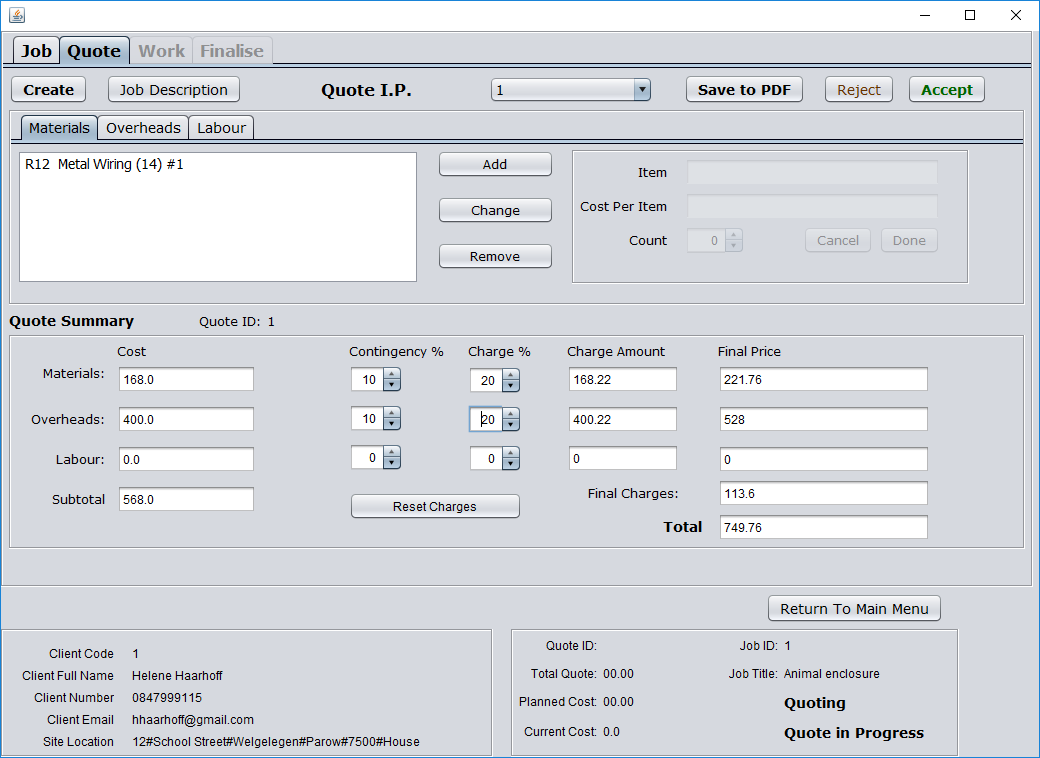
* + - 1. Select Create



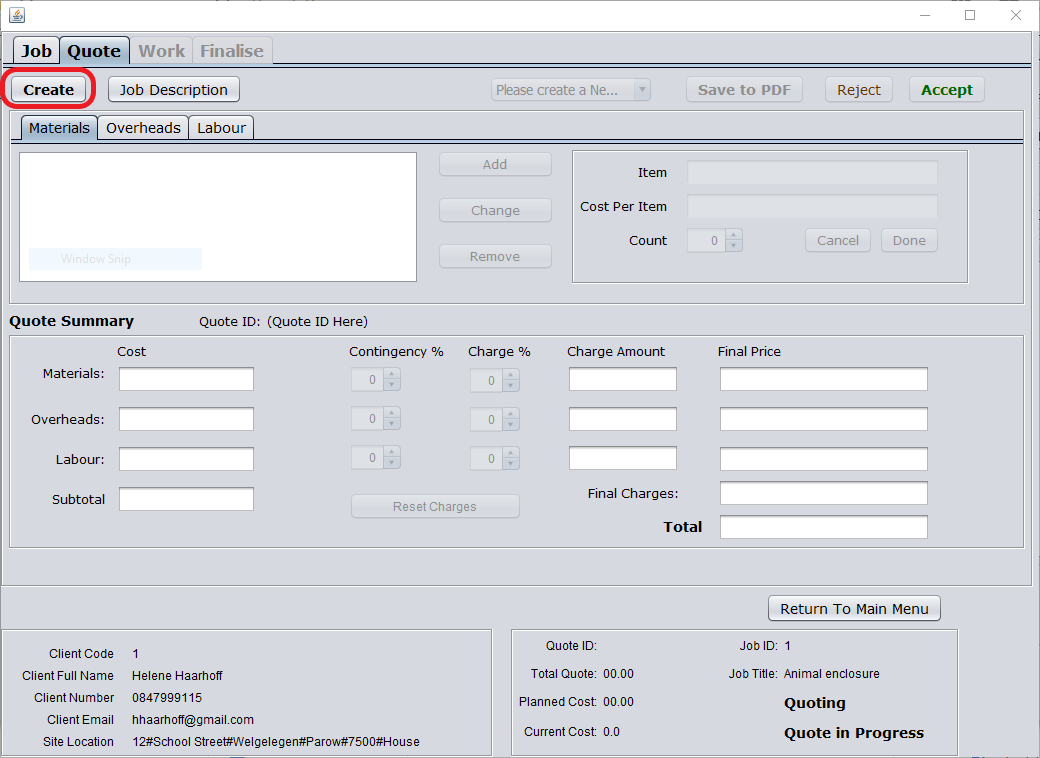
* + - 1. Insert details



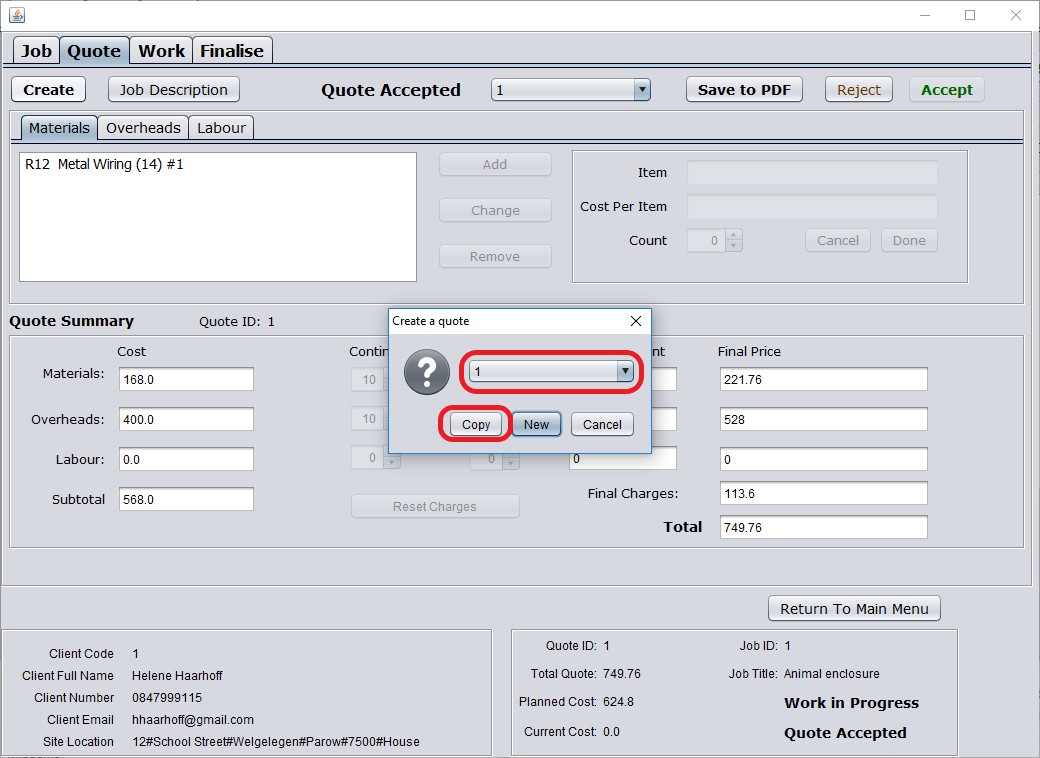
* + 1. Copy an existing quote
       1. Navigate to existing quote



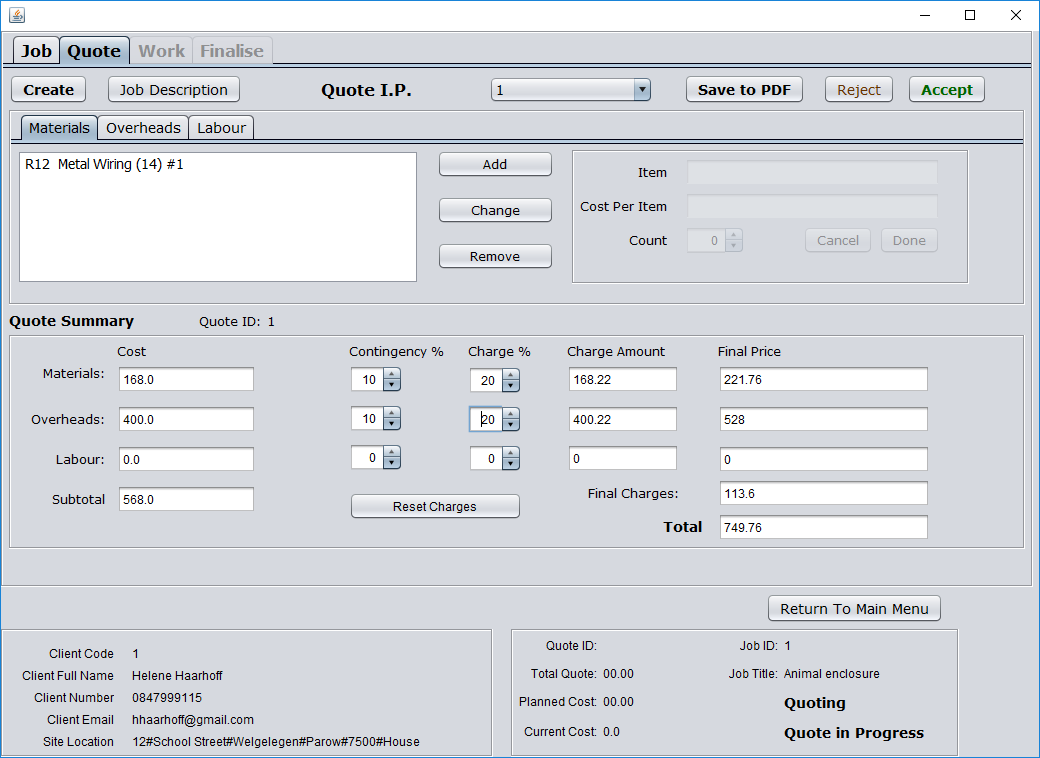
* + - 1. Select create



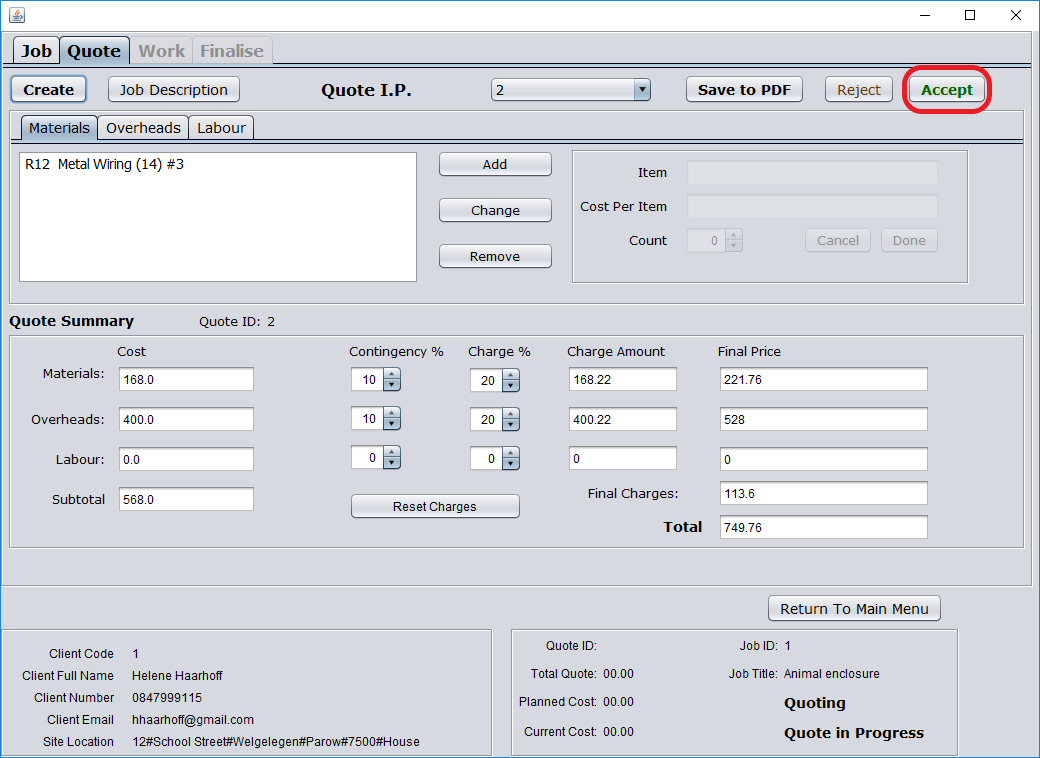
* + - 1. Select existing quote and select Copy



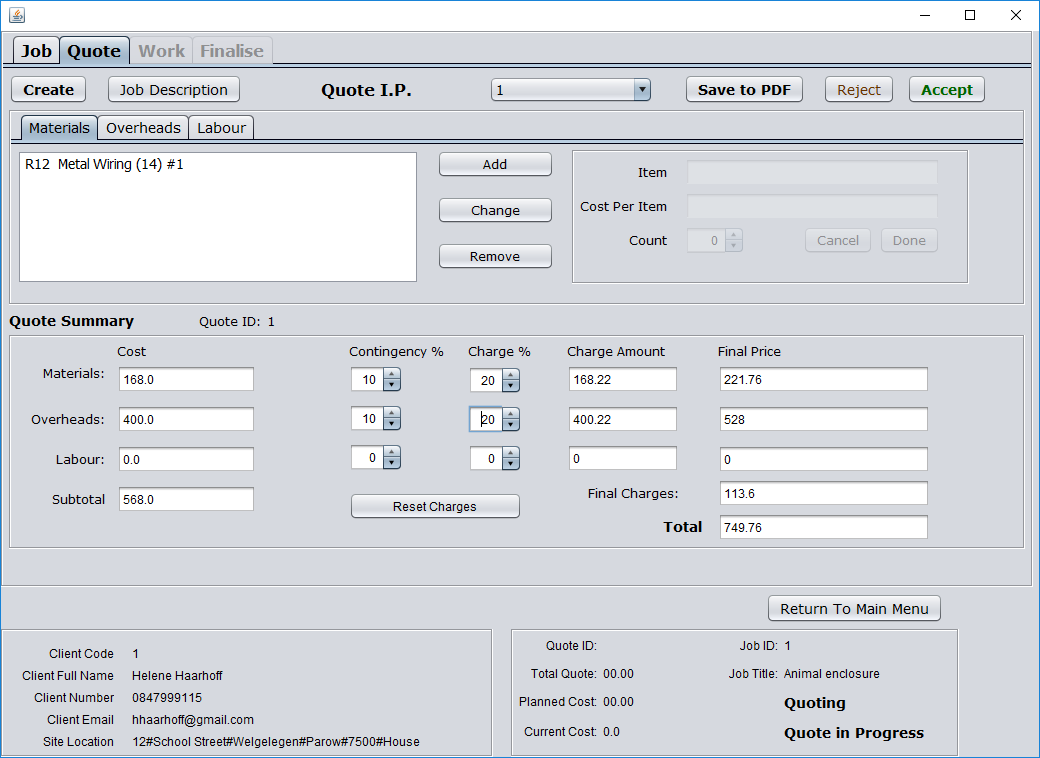
* + 1. Accept quote
       1. Navigate to existing quote



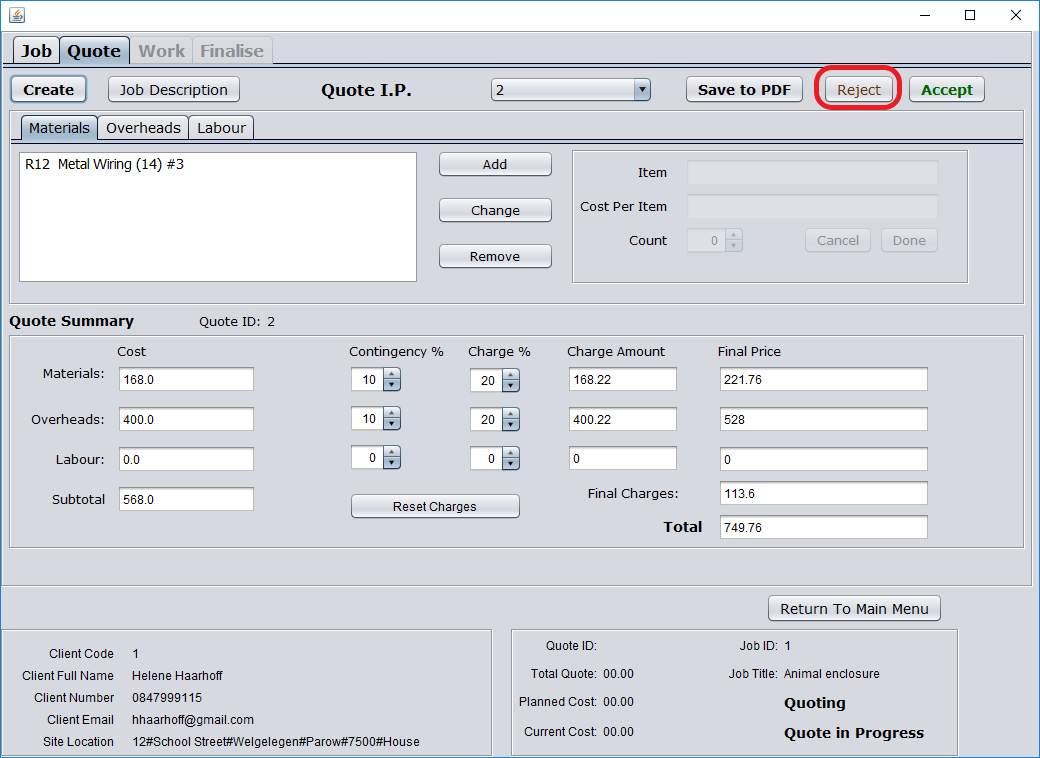
* + - 1. Select Accept



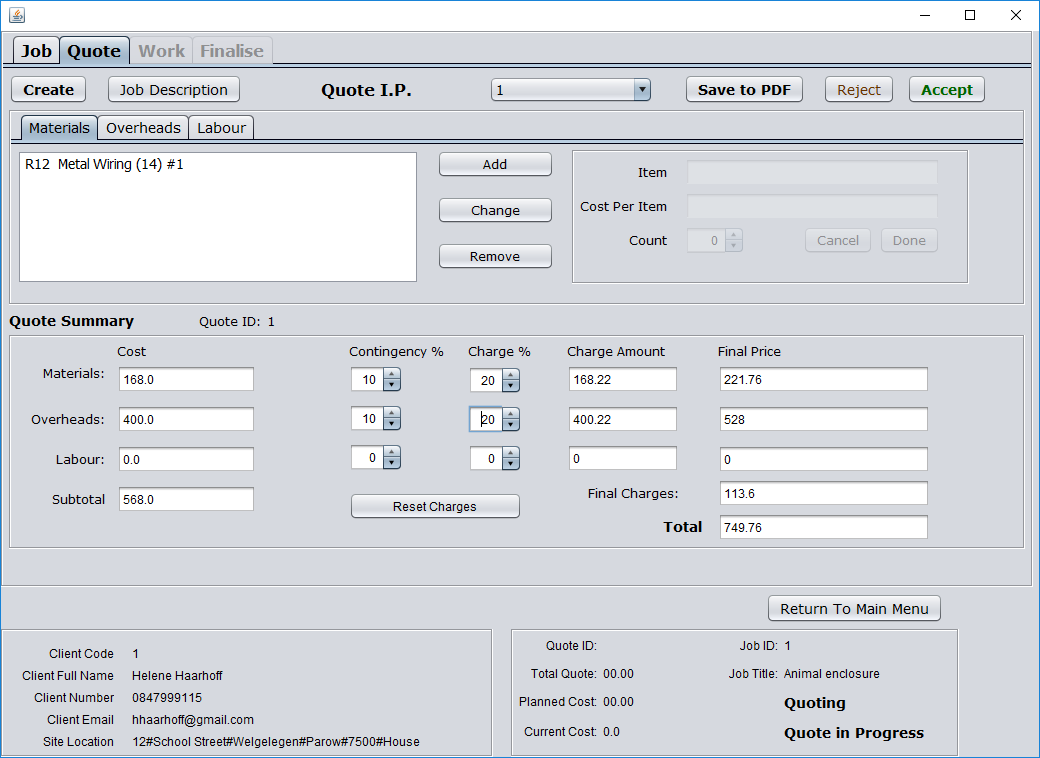
* + 1. Reject quote
       1. Navigate to existing quote



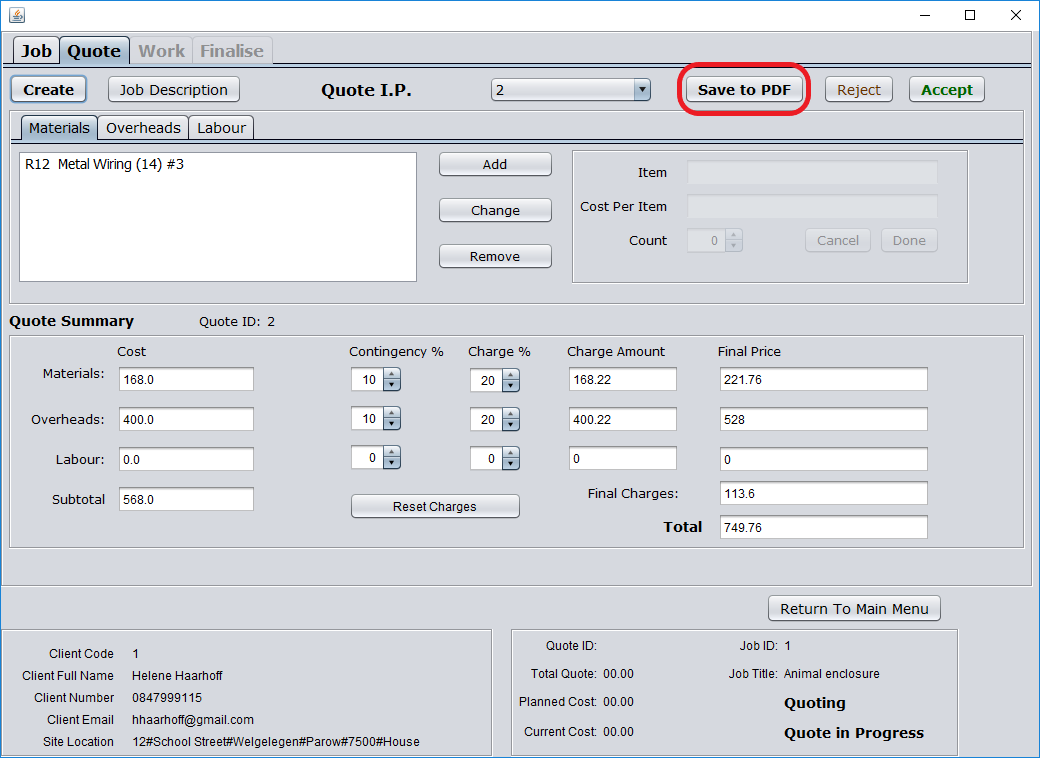
* + - 1. Select Reject



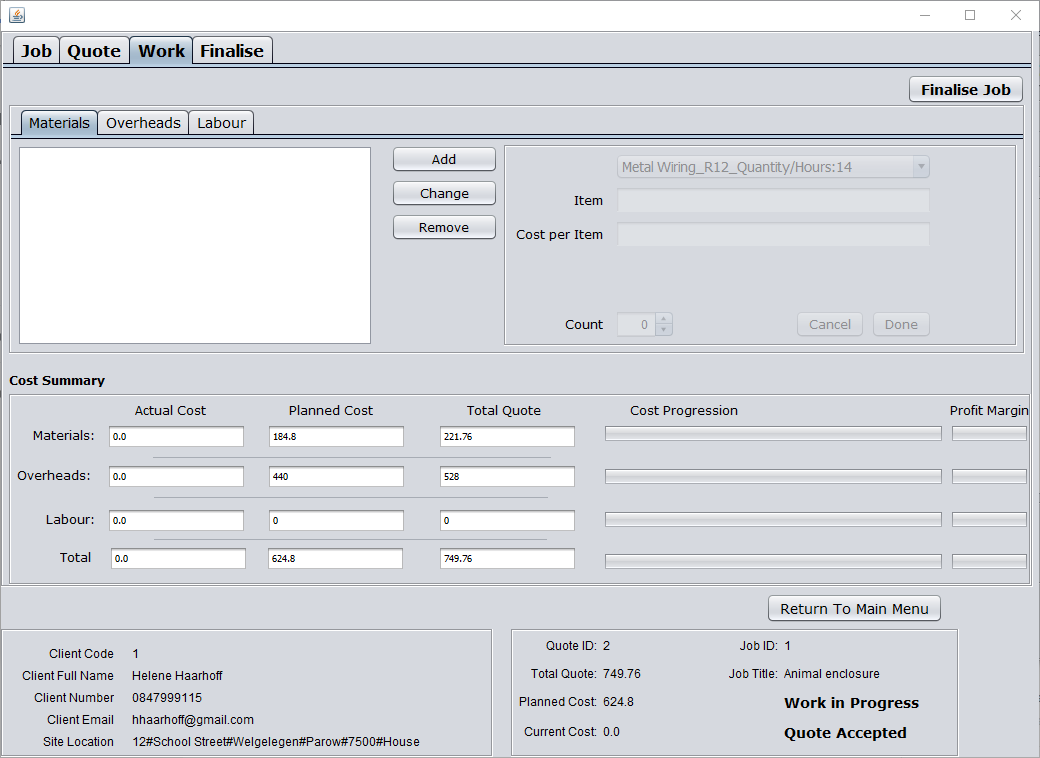
* + 1. Save quote to PDF
       1. Navigate to existing quote



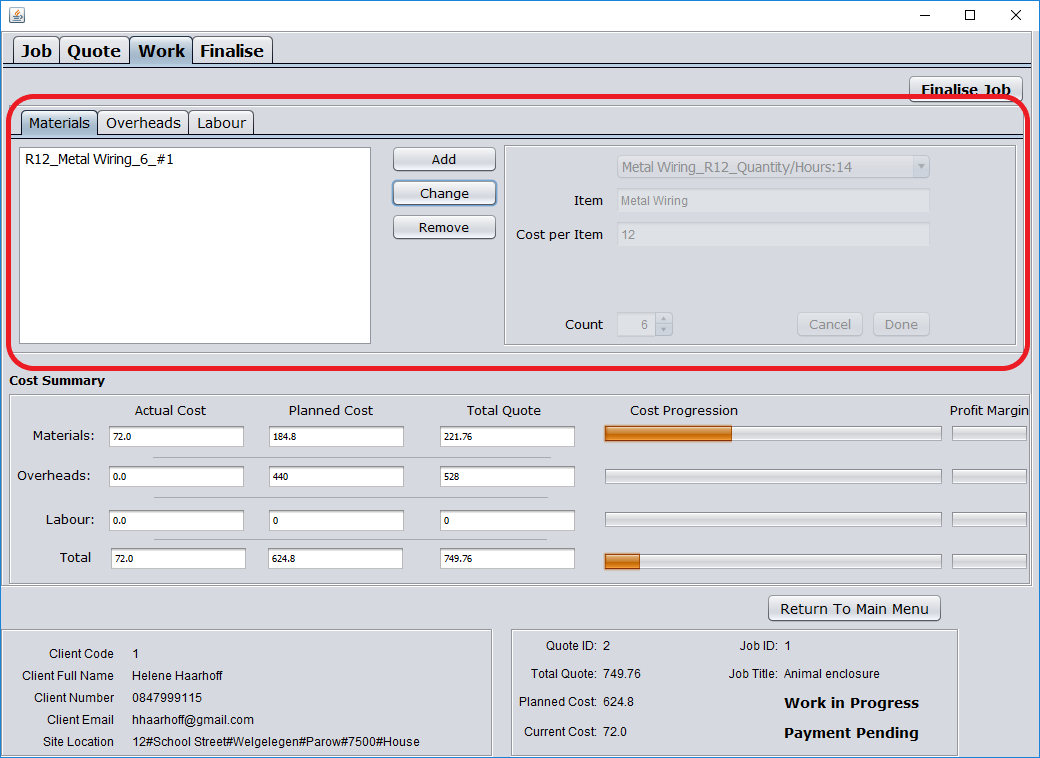
* + - 1. Select Save to PDF



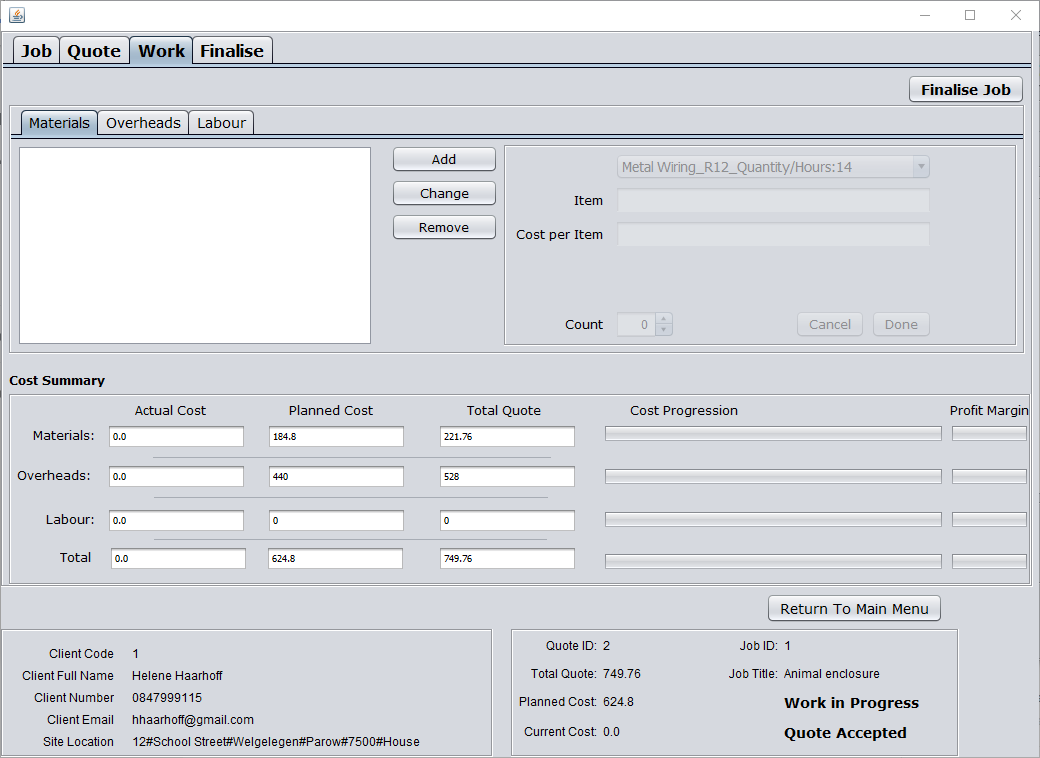
* + 1. Track work
       1. Navigate to work page



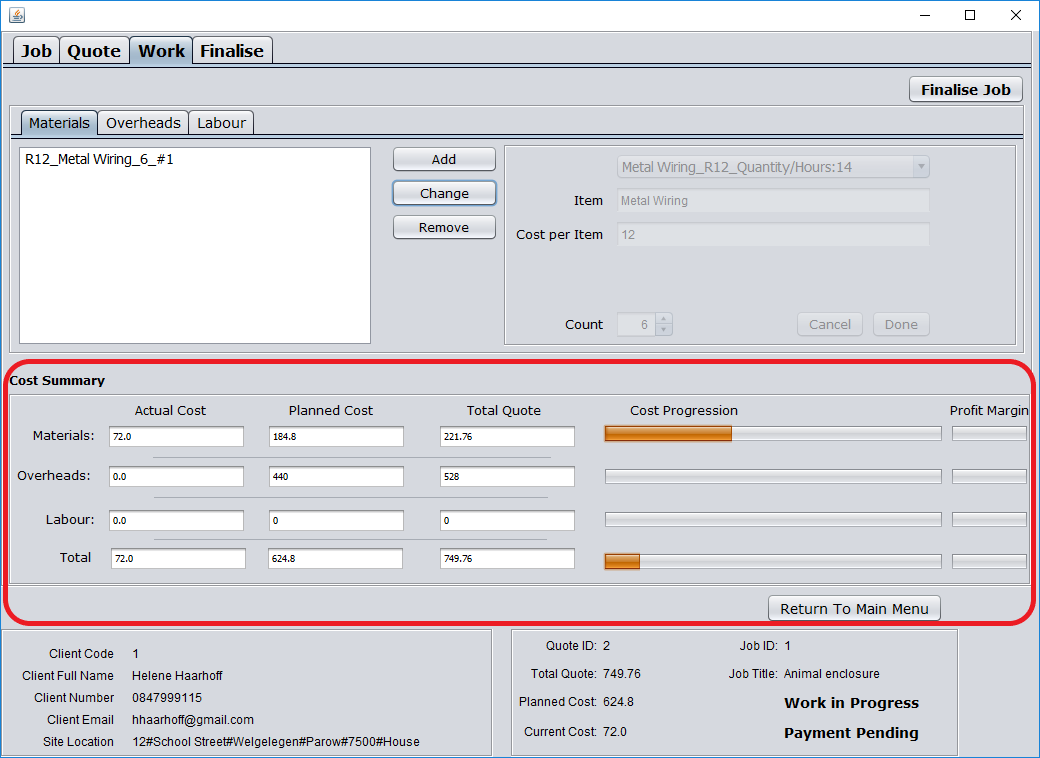
* + - 1. Add work



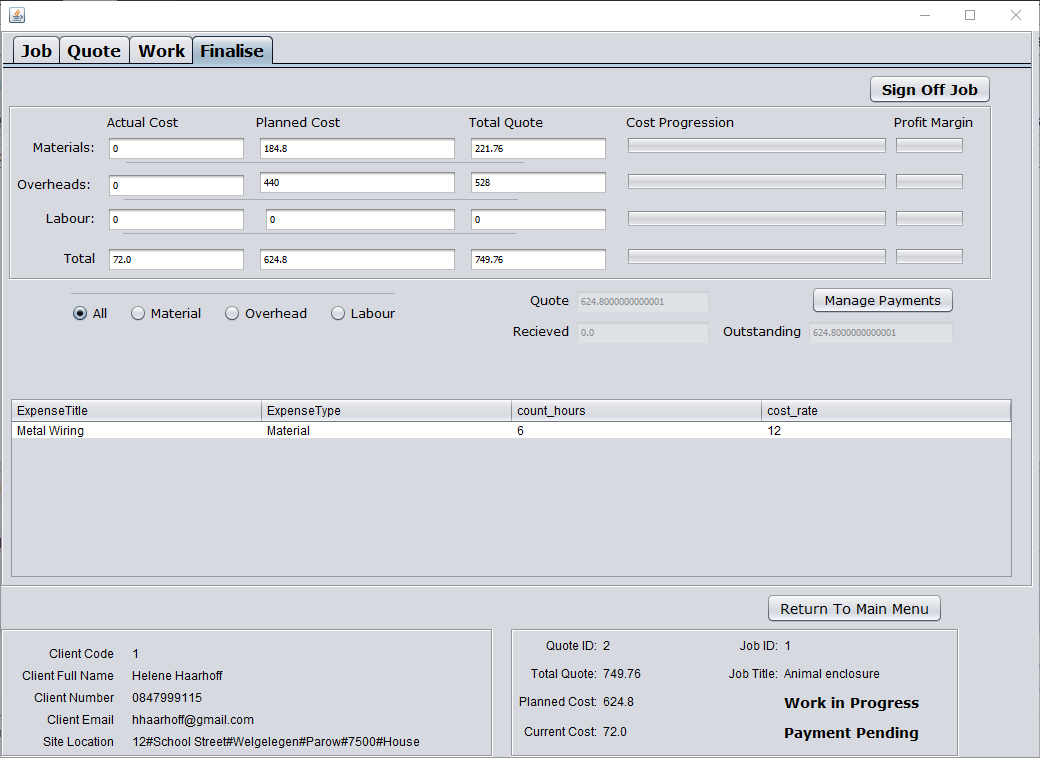
* + 1. Track spend
       1. Navigate to work page



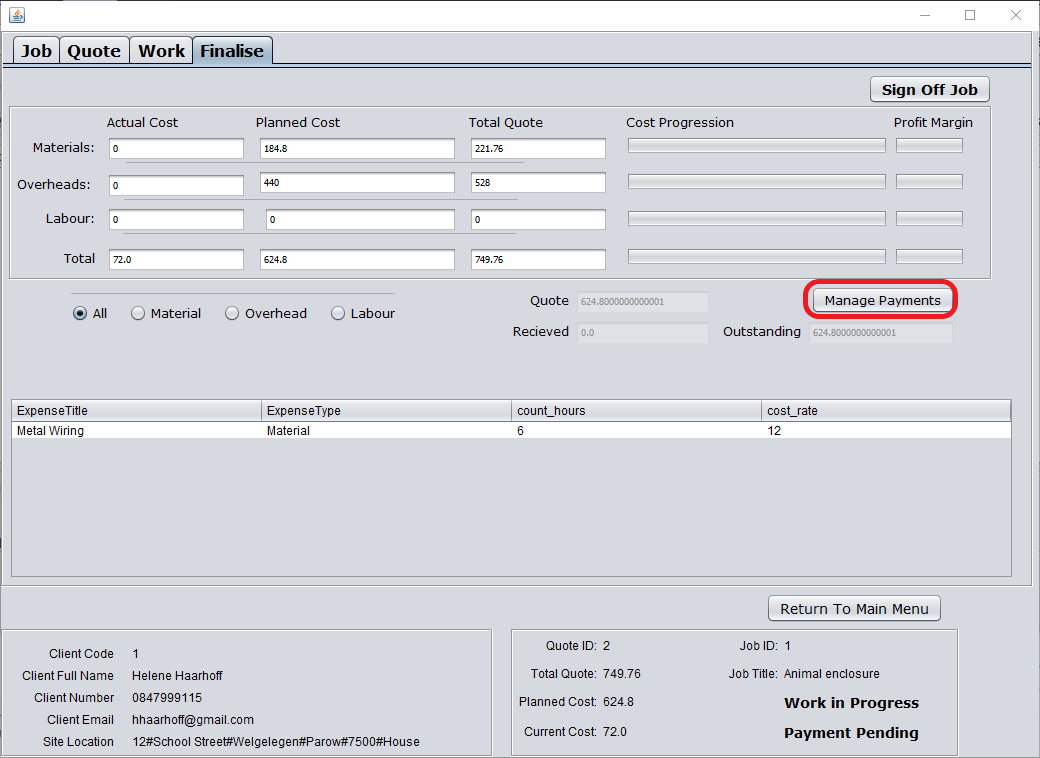
* + - 1. Track spend



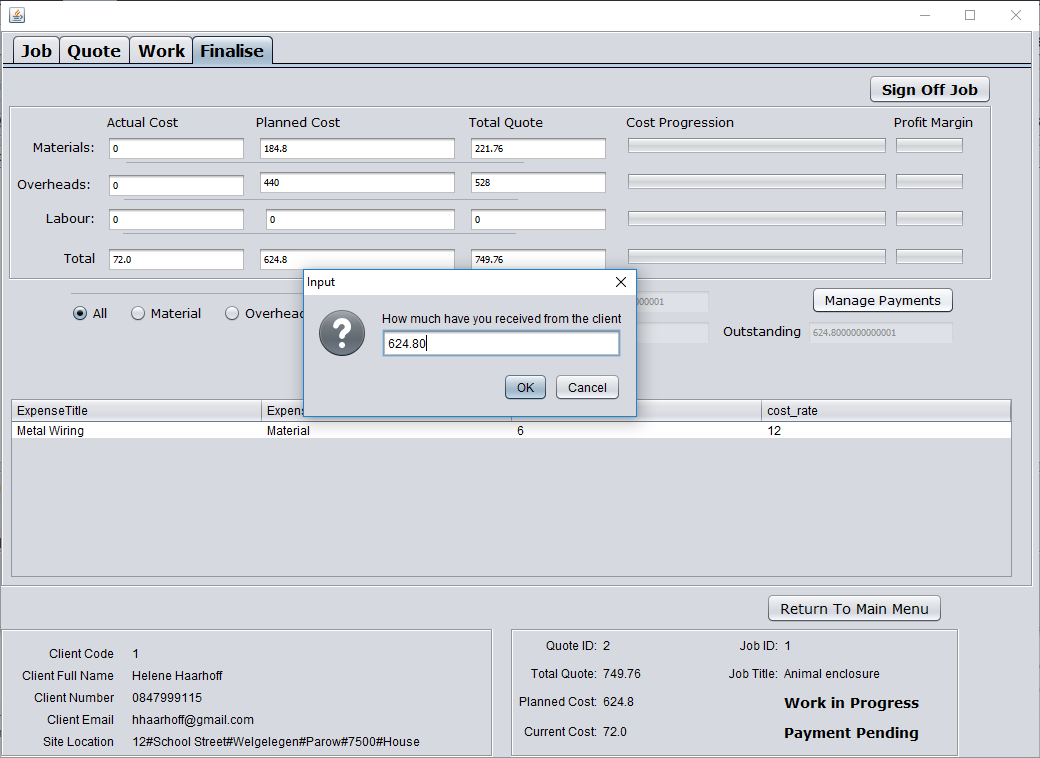
* + 1. Add payments made
       1. Navigate to finalise page



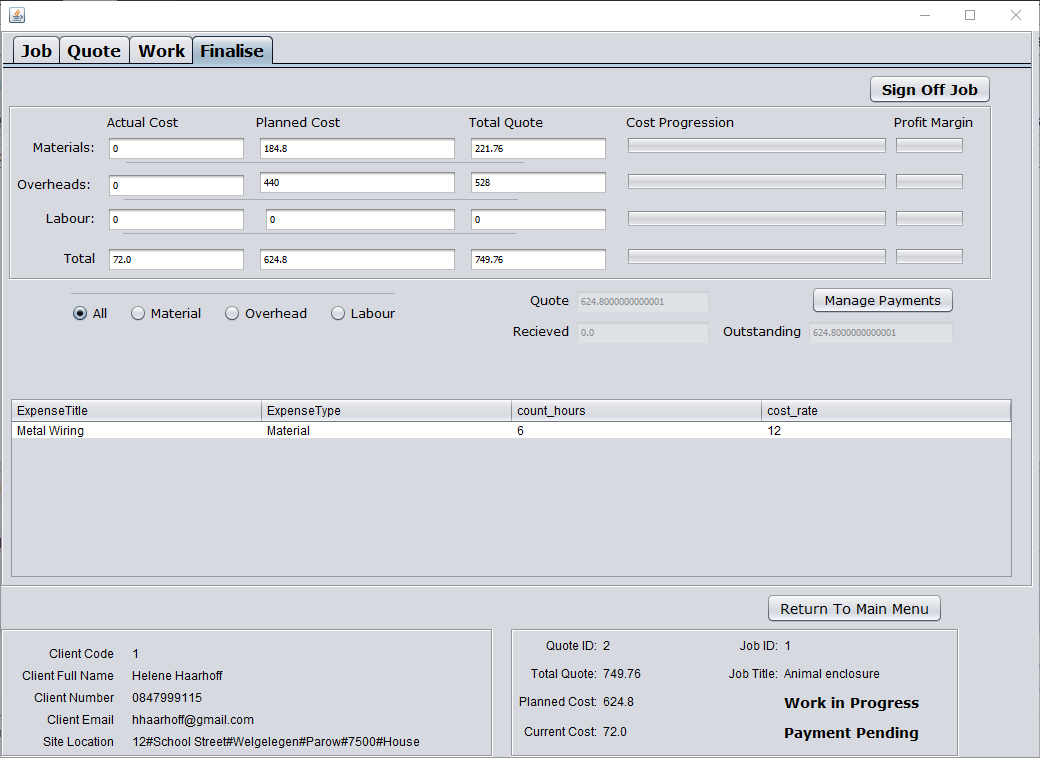
* + - 1. Select Manage Payments



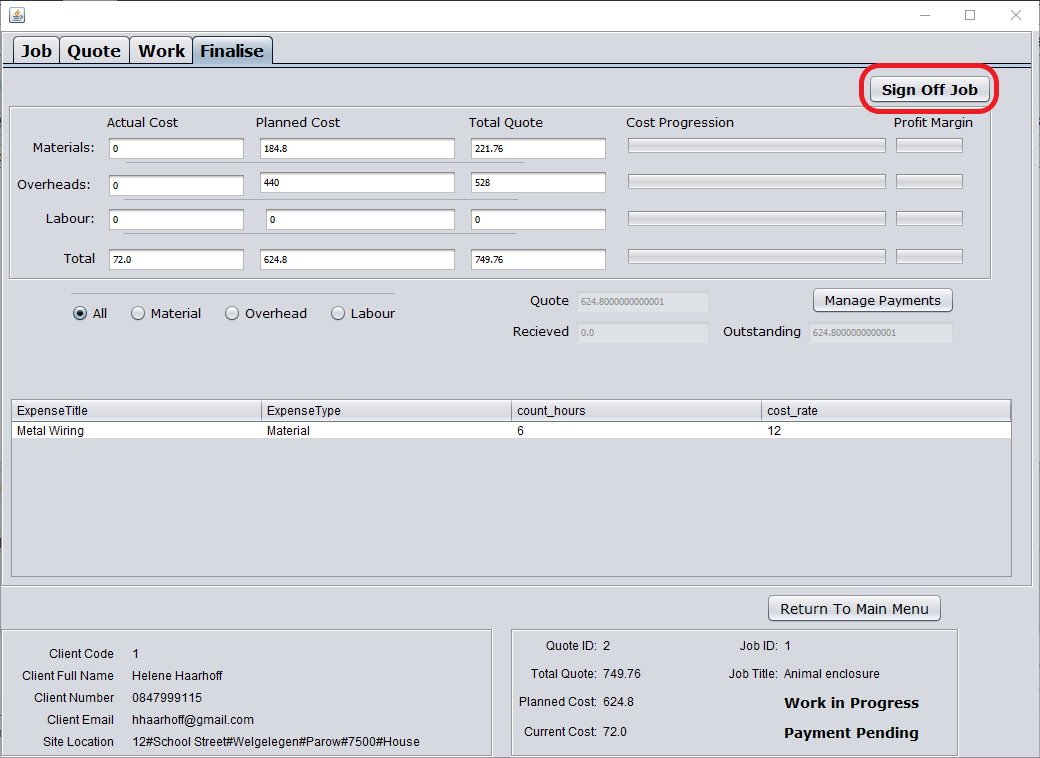
* + - 1. Enter payment amount



* + 1. Finalise job
       1. Navigate to finalise page



* + - 1. Select Sign Off Job



1. Deliverable 5: Evaluation report
   1. Introduction

This report evaluated the project in terms of project management. It discusses the final system and how it relates to the initial functional requirements set out by the customer, delves into the inner workings of the methodology used in the development planning of the project. The customer is reviewed in terms of his knowledge, availability and input into the final result. The group dynamics and team collaboration are discussed as well as explaining how time management was handled. Finally, each team member gives their opinion of the lessons learned in the completion of this project.

* 1. The final system and the customer’s requirements

|  |  |
| --- | --- |
| REQUIREMENT MET | REQUIREMENTS NOT MET |
| Store User data | Create schedules for job |
| List all jobs | Flag irregularities due to changes been made. |
| List all active jobs at the current time (or specific time) | Checkout of various equipment |
| Search jobs by identifier | Notify user about time or resource conflicts |
| Produce various report diagram to report (Graphs) | Store signed copies of each client’s indemnity form |
| Create Job |  |
| Change Job State and data (active, employees, completed, etc.) |  |
| Manage the job specification changes |  |
| Post job site inspection to ensure job is done |  |
| Manage employees on job |  |
| Store employee data (create, edit, remove) |  |
| Track employees hours worked and salary due |  |
| Manage and assign resources |  |
| Store quotes from suppliers |  |
| Budget estimated work hours for job |  |
| Compile a site report (also used for quotation) |  |
| Issue calculated quote for the client |  |
| Forecast income and expenses based on job input |  |
| Store/change client data |  |

As seen in the table above, most of the functional requirements were met. The main exception being **scheduling** which was deemed to be surplus to requirements. It should also be noted that the requirements that were not met were all rated as ‘could have’ according to MoSCoW prioritisation.

* 1. The methodology

It was decided that an iterative incremental development methodology would be used for this project. The initial development plan consisted of five activities, which upon completion would result in an iteration of the project being completed. These activities were logical design, interface design and development, system development, system testing, and documentation. Each activity was assigned to a team member with Ryan Ford acting a project manager and ensuring that all deadlines were met.

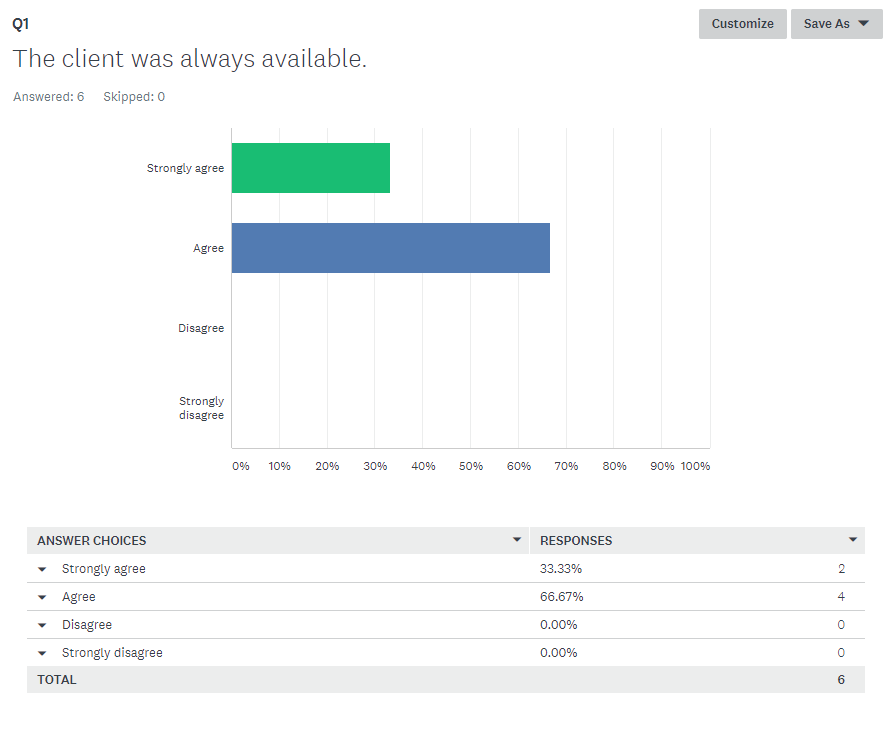
This development plan did not work as well in practice as it did in theory as it seemed more practical to complete the entire logical design of the project within the first iteration. The reason being that it would be easier to see the bigger picture of what the system should be and how each module of the system interacted within the entire system as a whole. The interface design and development followed suit and the team was making good progress, however when system development began, the flaws in this new development plan became apparent.

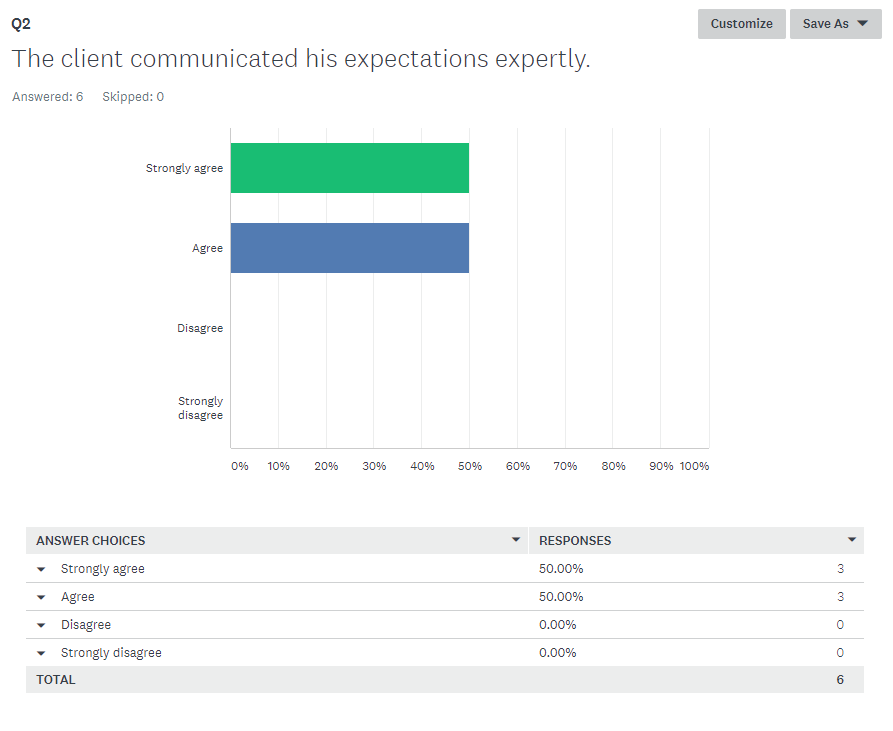
The system developer struggled to break the system up into modules and it became more and more difficult to track progress with constant changes being made to the interface, database and requirements. During this phase of uncertainty and lack of communication, the team felt that time was being wasted. A meeting was held and the shortcomings discussed resulting in the reallocation of resources and the forming of an improved development plan.

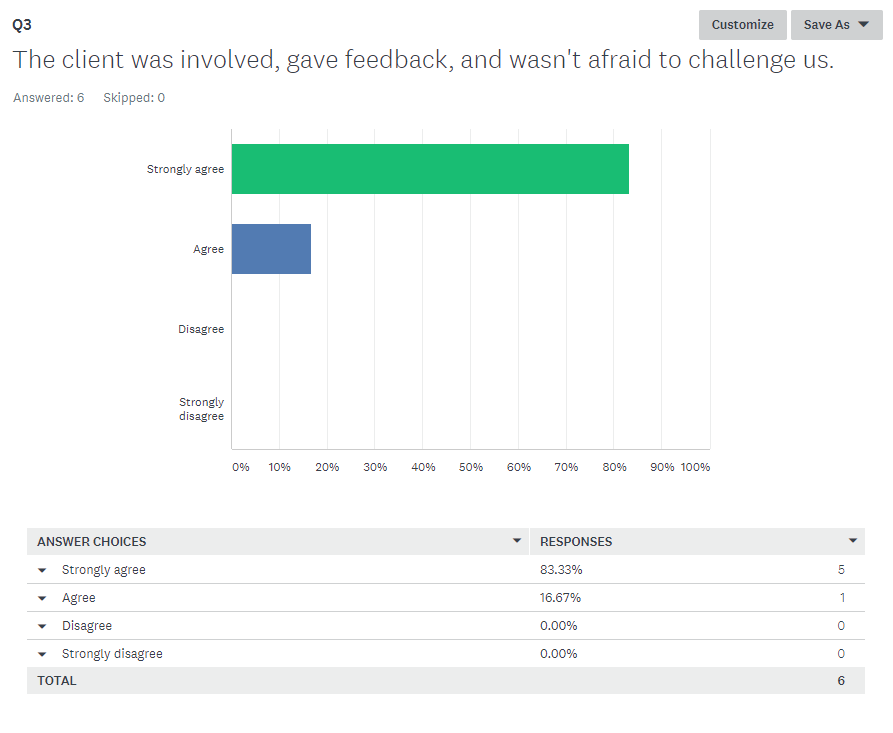
The system was broken up into modules with the system developer continuing to complete iterations as before, however the implementation of testing results (error fixing) were moved to other team members. This change resulted in a much better working cycle and ultimately ensured that the project was a success.

* 1. The customer

A survey was conducted to determine the general consensus surrounding the customer and the results are as follows:

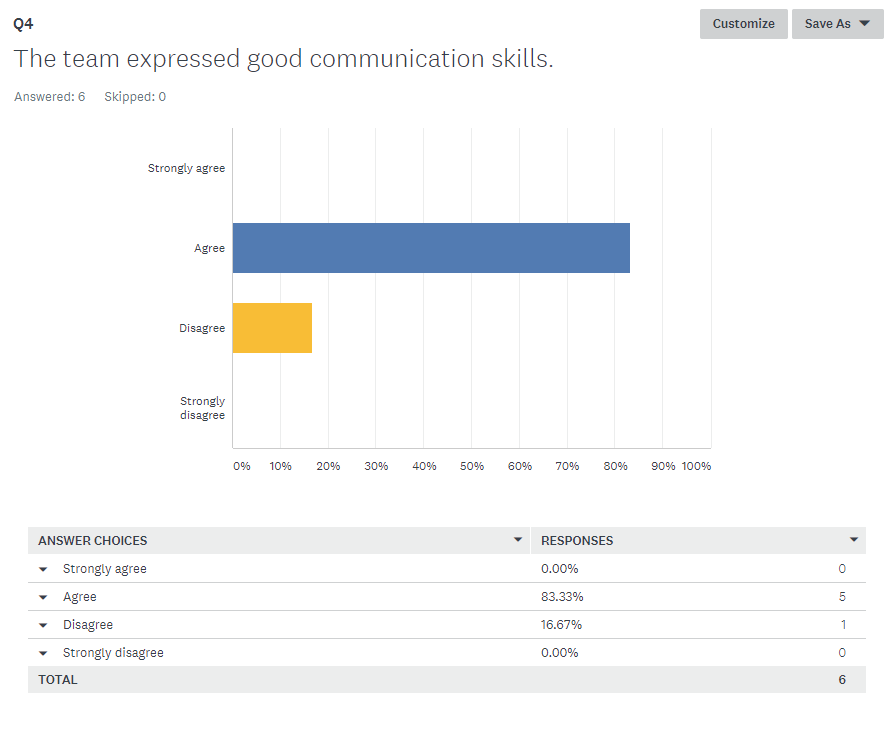


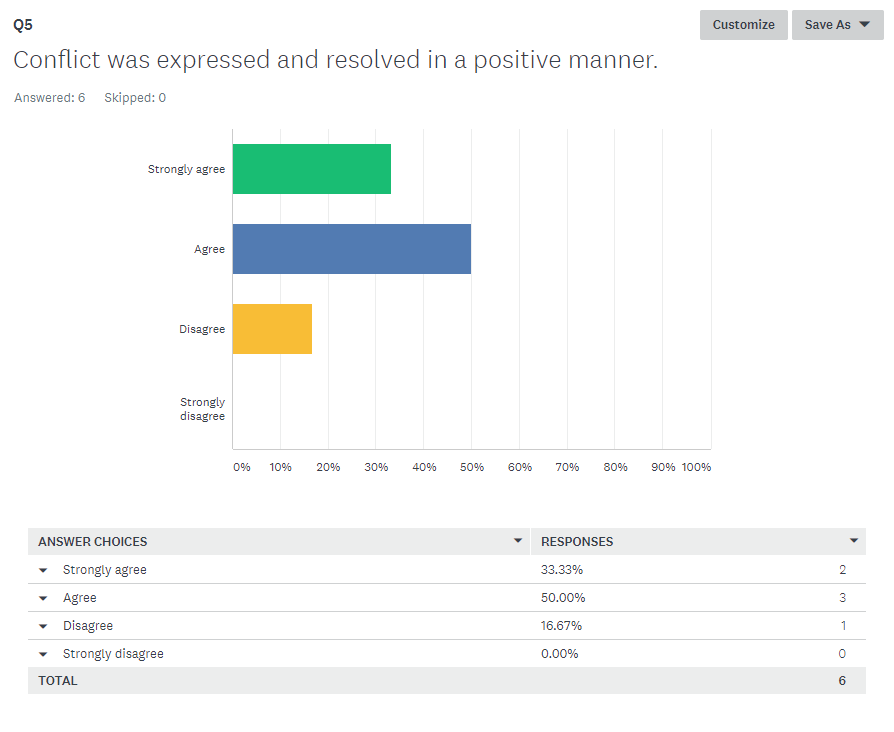


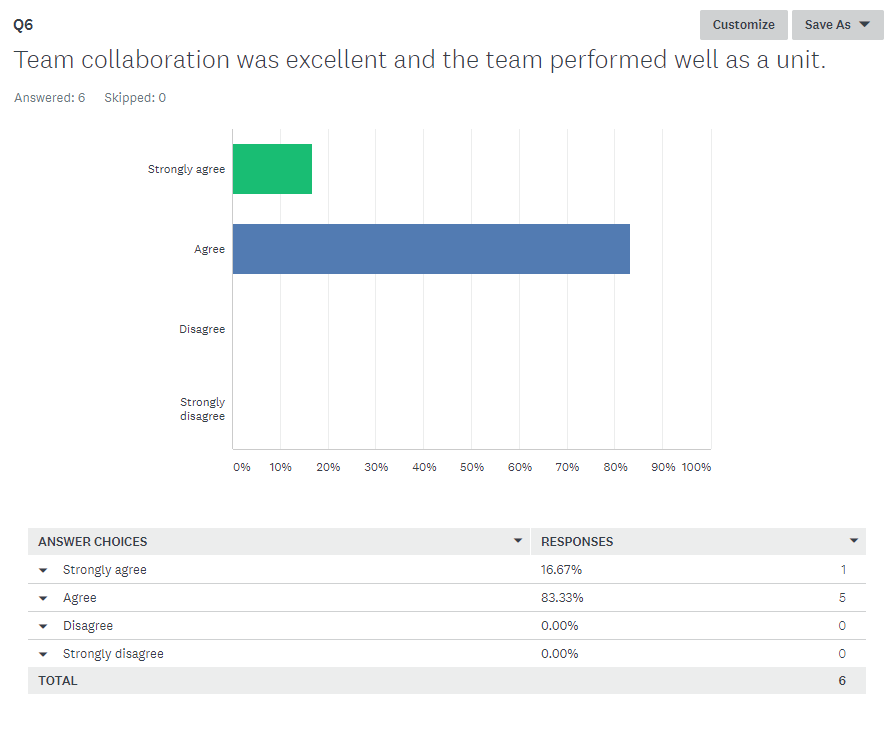


* 1. Group dynamics and team collaboration

A survey was conducted to determine the general consensus surrounding the group dynamics and team collaboration and the results are as follows:

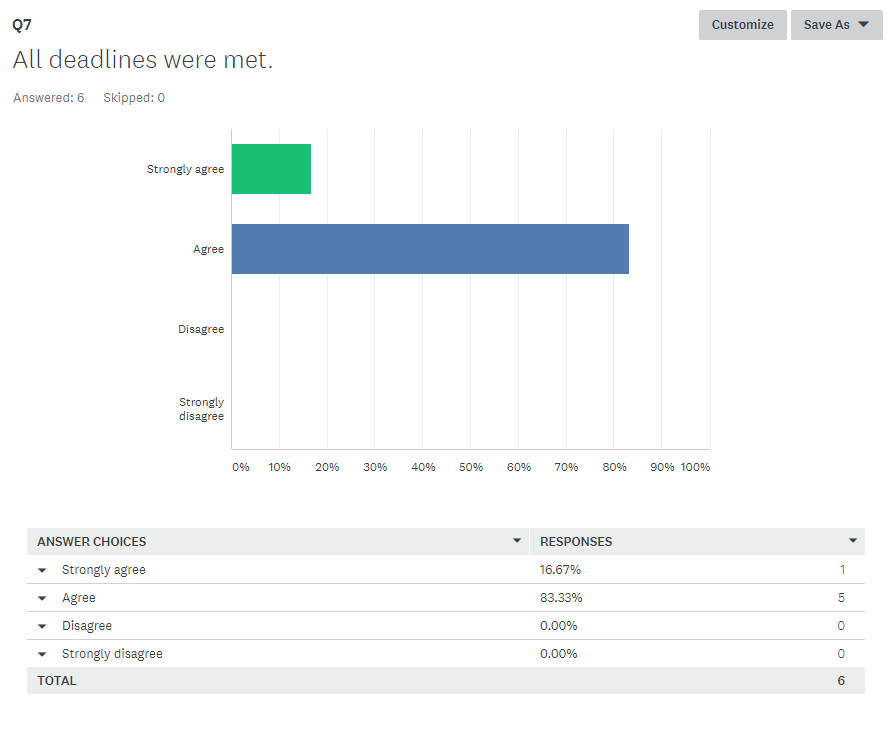


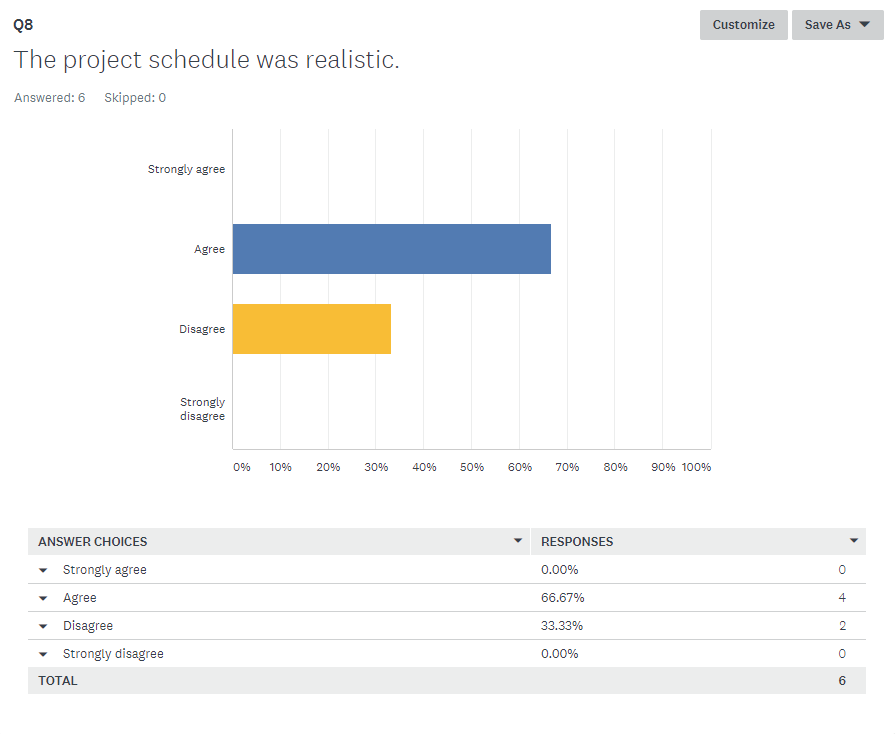


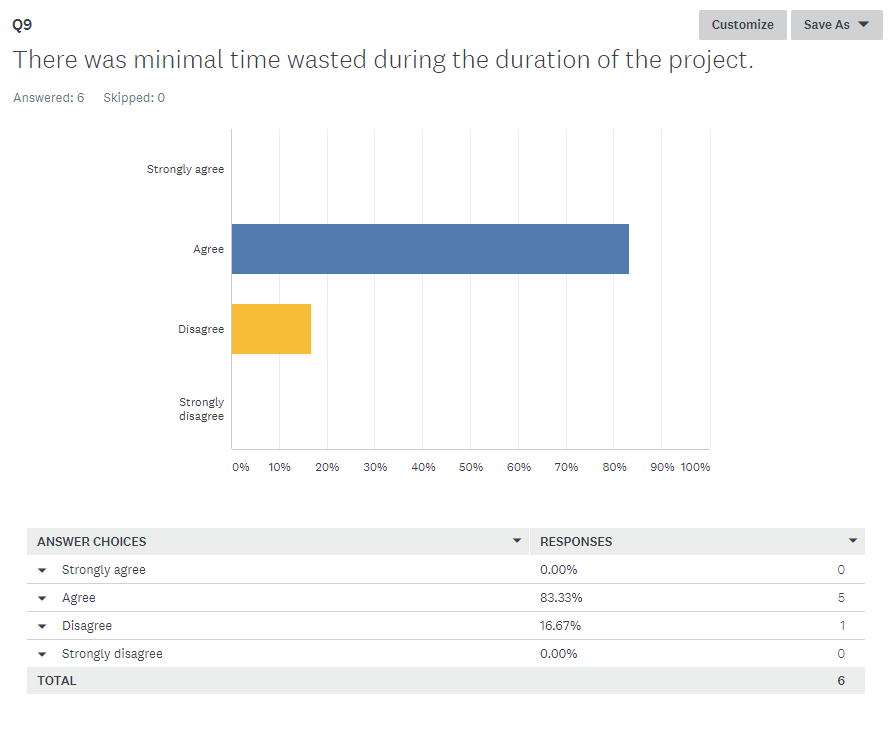


* 1. Time management

A survey was conducted to determine the general consensus surrounding time management and the results are as follows:







* 1. Lessons learned

*“The biggest lesson I learned was how to work as a team member and split up a large amount of work over a long period of time. I also learned how to use version control systems and how to communicate changes to my team members. As the lead programmer I also learned a lot about problem solving and how to meet client expectations when creating a system.”*

Adam Botha (2018)

*“I believe that the project was and still is a success, however it did provide valuable lessons. One of the key lessons learnt was to not be over confident in and take on more work than necessary and to start simple and build on a strong foundation.”*

Ryan Ford (2018)

*“I learned to manage time well, as well as discovered the dynamics within the team. It is refreshing to work with a competent team that is willing to work hard for themselves and for the team. I realised that we took on a larger project than initially anticipated, but I believe that we can deliver the solution in time.”*

James Kotze (2018)

“The main lesson I learned in the completion of this project is that breaking the project up into smaller tasks is essential to staying on track and not getting overwhelmed by the workload. I also learned that everything won't always go to plan and that it is impeccable to change something if it is not working and to manage that change properly.”

Dreyer Morkel (2018)

*“No comment.”*

Jonathan Prince (2018)

*“The main lessons I take away from this project - some not new, but still - are the following. If you work hard you can achieve anything. Many hands make light work. Embrace change and adapt your plans, but for that to work you need a good plan to start with. Don't be ignorant of the people in your group, listen to them and help them to help the project in ways they can do best. I really should take on a project that I develop incrementally; it is difficult to achieve the mind-set of an evolving system, mostly because you want to have a complete design. As stated earlier though, plan to change and change the plan. Having a good project manager who can resolve issues and douse the fire is really neat (thanks Ryan). Having each member playing to their strengths really is key; also doing what they are interested in. This ensures no one becomes bored as well as irritated as the end draws near.”*

Franco van Zyl (2018)

* 1. Conclusion

In conclusion the final system has been completed and received successfully by the customer. All of the functional requirements were met, with the exception of a job scheduling function which was deemed to be an unrealistic requirement in terms of what the project schedule would allow. The iterative incremental development methodology proven to be successful, however the implementation of this methodology was adapted various times throughout the completion of the project. The customer was excellent in his knowledge of how Management Information Systems (MIS) function and gave clear and concise feedback. Ultimately the team worked well as a unit and everyone championed the project until completion, helping each other where assistance was required and coming up with solutions rather than focussing on problems. The general consensus being that the team adapted well to the changes made in the system development life cycle and that the project was a success.

1. Customer sign-off

|  |  |  |
| --- | --- | --- |
| **Customer name and surname** | **Customer signature** | **Date** |
| Ricky Ford |  |  |
| **Group leader name and surname** | **Group leader signature** | **Date** |
| Ryan Ford |  |  |

1. Bibliography

DevNami (2016). *Create PDF Document with iTextPDF Java*. [video] Available at: https://www.youtube.com/watch?v=OnEkVhAVSjA [Accessed 2 Nov. 2018].

Farcic, V. 2014, 21 January. Software Development Models: Iterative and Incremental Development. [online] Available at: <https://technologyconversations.com/2104/01/21/software-development-models-iterative-and-incremental-development/> [Accessed: 20 March 2018]

Ghahrai, A. 2017, 10 August. Difference Between Iterative and Incremental Development in Agile. [online] Available at: <https://www.testingexcellence.com/difference-iterative-incemental-development-agile/> [Accessed: 20 March 2018]

Java.com. 2018. *Download Free Java Software*. [Online] Available at: https://java.com/en/download/ [Accessed: 3 Aug. 2018].

Jrsoftware.org. 2018. *Inno Setup Downloads*. [Online] Available at: http://www.jrsoftware.org/isdl.php [Accessed: 3 Aug. 2018].

Kowal, G. 2018. *Launch4j - Cross-platform Java executable wrapper*. [Online] Launch4j.sourceforge.net. Available at: http://launch4j.sourceforge.net/ [Accessed: 3 Aug. 2018].

Microsoft.com. 2018. *Windows 10 Enterprise OS: Focus on Empowering Your Business*. [Online] Available at: https://www.microsoft.com/en-us/windowsforbusiness/windows-10-enterprise [Accessed: 3 Aug. 2018].

Netbeans.org. 2018. *NetBeans IDE - Overview*. [online] Available at: https://netbeans.org/features/index.html [Accessed: 3 Aug. 2018].

Products.office.com. 2018. *Database Software and Applications | Microsoft Access*. [Online] Available at: https://products.office.com/en-za/access [Accessed: 3 Aug. 2018].

Rongala, A. (2015). *What is Black Box Testing: Advantages and Disadvantages.* [online] Invensis Technologies. Available at: https://www.invensis.net/blog/it/black-box-testing-advantages-disadvantages/ [Accessed 29 Jul. 2018].

Rongala, A. (2015). *What is White Box Software Testing: Advantages and Disadvantages*. [online] Invensis Technologies. Available at: https://www.invensis.net/blog/it/white-box-software-testing-advantages-disadvantages/ [Accessed 29 Jul. 2018].

Synapsestudios.com. (n.d.). *Phoenix, Arizona Web Development, Custom Software, and Application Design. Synapse Studios*. [online] Available at: https://www.synapsestudios.com/understand/being-a-great-client [Accessed 10 Oct. 2018].

Technopedia, 2018. Iterative and Incremental Development. [online] Available at: <https://www.technopedia.com/definition/25895/iterative-and-incremental-development> [Accessed: 20 March 2018]

Ucanaccess.sourceforge.net. 2018. *UCanAccess-A Pure Java JDBC Driver for Access*. [Online] Available at: http://ucanaccess.sourceforge.net/site.html [Accessed: 3 Aug. 2018].

Zafar, R. (2012). *What is software testing? What are the different types of testing*. [online] Codeproject.com. Available at: https://www.codeproject.com/Tips/351122/What-is-software-testing-What-are-the-different-ty [Accessed 29 Jul. 2018].