Project Plan for a New CRM

Contents

Use-Case Breakdown into Tasks

- **A1.1: Add sales Owner role to account**

Tasks
1. Create Button to Navigate user to Account Page 2. Create User Interface for Account Page which will output all the accounts that admin is able to select. 3. Create a button to add role for each account 4. Create class for Sales Owner object and assign attributes for the Sales owner Object Functionality 5. Create Inheritance for one Sales Owner role per account. 6. Create Drop down menu that outputs the roles 7. Create Button to confirm change and let administrator add role to the account. 8. Create User interface to output feedback to the user after confirmation of the change. 9. Create functionality to navigate to the account page upon adding the role to the account. 10. Run tests on account information, checking that the Sales Owner Role has been Added
Dependencies: None (from the use cases listed). There is a dependency, however, on the administrator being logged in and the creation of the administrator object which can add roles to accounts.

- **A1.2: Remove sales owner role from account**

Tasks
1. Create logic to check if the account already has a role. 2. Create Button to delete role on the account if the account has a role 3. Create user interface for the remove role pop up and create output which will prompt the user to confirm the change they have just made. 4. Create Functionality to delete Sales Owner Attribute from the account upon confirmation from the user. 5. Create functionality to navigate to the account page upon deleting the Sales Owner role from the account. 6. Create and run test to check that the Sales Owner Role has been removed.
Dependencies
- Use Case A1.1: a sales Owner role must be added to the account for this use case to be satisfied.

- **A2.1: Entering a new lead**

Tasks
1. Create database table for pipeline to store all the lead information in it and be able to add new entries for new leads. 2. Create an object for the lead and all the lead details as attributes for the lead object 3. Create a button to allow the Sales Owner/Administrator to select the pipeline tab.

4. Create User interface for the pipeline tab and link it to the button so the user navigates to the pipeline tab once the button is pressed.
5. Create button to add new lead
6. Create user interface for the add new lead pop up and link to button so the pop up appears once the user clicks the button.
7. Create a form for users to fill in the lead details. The details of the form for the user to add regarding the leads can be found in the CRM requirements.
8. Create a button to submit form and confirm details that the user has filled in.
9. Create validation for the form, so if admin/Sales Owner has missed certain details from the form it will prompt them to fill in those details and prevent submission.
10. Create functionality to add the lead into the database table for the pipeline. Assign this method to the Sales Owner/Administrator object.
11. Create database table to store all events then log this event onto the database once the lead has been added.
12. Link account page to the submit form button. Create logic so upon confirmation, once the lead has been added and everything is correct it will redirect to the accounts page.
13. Run test to check that the lead has been added onto the database

Dependencies

- Use Case A1.1: a sales Owner role must be added to the account as only sales Owners/administrators will be able to add new leads.

- **A2.2: Update lead information**

Tasks

1. Create a button for the user to select the lead
2. Create a button to edit lead information
3. Create UI for edit lead pop-up
4. Create an input form for the user to input new values for the information that they want to edit.
5. Create Save changes button to confirm changes
6. Create functionality to assign new values to the attributes that whatever the user has edited in the lead object. Assign this method to sales Owner/administrator object.
7. Implement the functionality to edit values once the save changes button has been pressed.
8. Create event for the editing of the lead info and log onto system events DB.
9. Link pipeline tab to save changes button so user navigates to the pipeline tab once the button has been clicked.
10. Create and run unit tests to make sure that the new values are up to date with what the user has edited.

Dependencies

- Use Case A1.1: a sales owner role must be added to the account as only sales Owners/administrators will be able to update lead information.
- Use Case A2.1: A lead must have been added for this use case to be satisfied. You can't update a lead if it has not been added.

- **A2.3: Change pipeline stage of a lead**

Tasks
1. Create attributes of the different pipeline stages which will be stored in the lead object created. 2. Create UI that outputs the stages that the lead is currently on within the pipeline tab 3. Create a button on the lead that the user can click and drag. 4. Create functionality to change the positioning of the lead once it has been dragged. Assign this method only for sales owners/administrator objects 5. Create functionality so depending on where the user has dragged the lead to, it will update the stage of the lead. This method is only assigned to sales owner/administrator object 6. Create an event showing that the pipeline stage of the lead has been updated 7. Log this event onto the events database by adding a new entry. 8. Create and run tests to make sure that the current pipeline stage is up to date with what the sales owner/administrator has edited.
Dependencies
- Use Case A1.1: a sales owner role must be added to the account as only sales Owners/administrators will be able to change the pipeline stage of a lead. . - Use Case A2.1: A lead must have been added for this use case to be satisfied. You can't change the pipeline stage of a lead if it has not been added.

- **A2.4: View edit history of lead**

Tasks
1. Create a view edit history button that is output once the user clicks the "Select lead" button. 2. Create User Interface for edit history pop-up that is outputted once the user clicks on the view edit history button 3. Create and run test to check that the edit history has been outputted and has been viewed.
Dependencies
- Use Case A2.1: A lead must have been added for this use case to be satisfied. You can't view the edit history of a lead if it hasn't been added. - Use Case A.2.2: A lead must have been edited to be able view the edit history. You can't view the edited history of a lead that has not been edited.

- **A2.5 Toggle view mode in pipeline**

Tasks
1. Check that user class can have active account 2. Link toggle view code to verification of user having active account 3. Write code to verify the user being in the pipeline tab and link to toggle view code 4. Add a "Toggle view mode" button to the pipeline tab 5. Design UI style for each view mode

6. Add code to turn data into UI for each view mode 7. Add code for button to control which visualization code is used 8. Write and run tests on visualization code
Dependencies
- User class, login information DB, user verification code - Use Class A2.1: lead object, DB of leads, button to select pipeline tab, general pipeline tab UI

- **A2.6 Search for query in leads**

Task
1. Check that user class can have active account 2. Link search code to verification of user having active account 3. Write code to verify the user being in the SalesRock Module and link to search code 4. Add a “search” text field to the pipeline UI 5. Write code to query pipeline DB using the text in the search field 6. Link queried data to pipeline visualization 7. Write and run tests on user verification 8. Write and run tests on pipeline redirection, DB querying, and data visualization
Dependencies
- User class, login information DB, user verification code - Use Class A2.1: lead object, DB of leads, button to select pipeline tab, general pipeline tab UI

- **A3.1 Generate invoice and send to administrator system**

Tasks
1. Create sales invoice class 2. Create invoices DB 3. Check that user class can have sales owner/administrator roles 4. Link code to verification that user is a sales owner/administrator and is logged in 5. Link code to verification that user is in the SalesRock module 6. Add button “Generate invoice” that appears when lead is selected 7. Design “new invoice” popup with relevant fields to fill in and a “Generate invoice” button 8. Write code to generate popup upon button click 9. Add code to take popup fields and generate an invoice 10. Add code to send invoice to administrator system 11. Design user feedback, such as a “successfully generated” popup 12. Write code to display feedback and close “new invoice” popup upon successful generation 13. Write and run tests for popups and buttons 14. Write and run tests for invoice generation and sending to administrator system
Dependencies
- User class, login information DB, user verification code - Use Case A1.1: Sales Owner/Administrator roles - Use Class A2.1: lead object, DB of leads, button to select pipeline tab, general pipeline UI

- **A3.2 Change invoice status**

Tasks
1. Check that user class can have administrator role 2. Link code to verification that user is an administrator and logged in 3. Link code to verification that user is in SalesRock module 4. Design error handling in the case of nonexistent invoice 5. Add code to verify that invoice already exists, with error handling 6. Add an “invoices” tab to SalesRock UI 7. Design invoices tab UI 8. Add code to create invoices table from invoices DB 9. Add code to create invoices tab UI upon button click 10. Add code to make invoice display “Status” dropdown menu on selection 11. Add code to change invoice status in DB when a status on the menu is selected 12. Add code to update UI on DB changes 13. Write and run tests on user verification 14. Write and run tests on status changing
Dependencies
- User class, login information DB, user verification code - Use Case A1.1: Administrator role - Use Case A3.1: Invoice DB

- **A3.3 Download invoice**

Tasks
1. Check that user class can have sales owner/administrator role and can be active 2. Link code to verification that user is a sales owner/administrator, is logged in, and has an active account 3. Link code to verification that user is in SalesRock module 4. Add “download” button to invoice 5. Design PDF version of invoice 6. Add code to turn the information in invoice class into PDF 7. Add code to prompt browser to download invoice as PDF 8. Design user feedback on download, such as a notification 9. Add code to provide feedback on download 10. Write and run tests on user verification 11. Write and run tests on downloading
Dependencies
- User class, login information DB, user verification code - A3.1 invoice DB, invoice class - A3.2 invoice UI, button to redirect to invoice tab

- **A4.1 Change filter of an active visualization**

Tasks
1. Check that user class can be active

2. Link code to verification that user is logged in, on SalesRock visualization tab, and has an active account 3. Add code to check that there is an active visualization 4. Add “Change filter” button to visualization tab 5. Design “change filter” popup with date range, stage, and project fee filters 6. Add code to display popup upon button click 7. Design input field types for filters 8. Add code to validate filter inputs 9. Add “confirm change” button 10. Add code to set new filter 11. Add code to query visualization DB with new filter 12. Add code to reload visualization with new data 13. Write and run tests on user verification 14. Write and run tests on filter setting 15. Write and run tests on DB querying and visualization loading
Dependencies
- User class, login information DB, user verification code - Visualization DB, visualization class - Use Case A4.2: visualization tab, KPI visualization

- **A.4.2 Visualize predefined KPI profile**

Task breakdown
1. Check sure user is logged into an active account that is either an administrator, sales owner, or employee 2. Create visualization tab in the SalesRock module 3. Create initial prompt to view profiles and build out the option to view predefined KPI profiles. These KPI profiles should be stored in a database. Write code to get entry from the database. An entry in the database contains: a KPI visualization and a unique name. 4. Create a KPI visualization. These visualizations are stored in the predefined KPI profile database. A KPI visualization contains: a set of filters, target-data for the y-axis, range-data for the x-axis, a set of graphs, a target-objective 5. Display the visualizations in accordance with the KPI profile. This means designing the page such that all aspects of the visualization are well presented for the user. 6. Write and administer appropriate tests on user verification 7. Write and administer appropriate tests on the visualizations 8. Write and administer appropriate tests on the KPI profile database
Dependencies
A.4.1: A KPI visualization’s filters must be built before they are incorporated into a predefined KPI profile The user system must be created so that only administrators, sales owners, and employees can access this.

- **A.4.3 Add custom profile**

Task breakdown
1. Check sure user is logged into an active account that is either an administrator or sales owner 2. In the visualizations tab, add an option to create a custom profile onto the initial prompt 3. Create a “configuration tool” to create a custom profile. This creates a new entry that should be added to the KPI profile database. The configuration tool will prompt the user to Fill out a unique name, add filters, select different graphs, select data for the graphs, choose if the new profile should be added to the “custom profiles” set 4. Create a “Save” button that gives feedback on the profile, such as if the name is unique. 5. Display the new KPI profile visualizations in the same way as a predefined KPI profile from use case A.4.2 6. Write and administer appropriate tests on user verification 7. Write and administer appropriate tests on adding a new entry to the KPI profile 8. Write and administer appropriate tests on displaying the profile
Dependencies
A.4.2: - Visualization tab is created - An initial prompt appears - Database is created - Visualization page is displayed The user system must be created so that only administrators and sales owners can access this.

- **A.4.4 Visualize custom profile**

Task breakdown
1. Check sure user is logged into an active account that is either an administrator or sales owner 2. In the visualizations tab, add an option to select a custom profile onto the initial prompt 3. Create a modal that allows the user to select one of the existing custom profiles. 4. Display the new KPI profile visualizations in the same way as a predefined KPI profile from use case A.4.2 5. Write and administer appropriate tests on user verification 6. Write and administer appropriate tests on the various profile selection through visualization
Dependencies
A.4.2: - Visualization tab is created - An initial prompt appears - Database is created - Visualization page is displayed A.4.3 - Custom profiles have been created - Completed custom profile is displayed

The user system must be created so that only administrators and sales owners can access this.

- **A.4.5 Export a visualization**

Task breakdown
1. Check sure user is logged into an active account that is either an administrator, sales owner, or employee 2. On the visualizations display, create a “Download as” button that pulls up a dropdown menu. 3. Download the visualization in the selected format through the browser 4. Write and administer appropriate tests on user verification
Dependencies
A.4.2: - Visualization tab is created - Database is created - Visualization page is displayed The user system must be created so that only administrators, sales owners, and employees can access this.

- **A.5.1 Adding a customer**

Task breakdown
1. Check sure user is logged into an active account that is either an administrator or sales owner 2. Create customers tab and a references tab in the SalesRock module 3. Create a Customer database and display entries upon selecting the customers tab 4. Create a References database and display entries upon selecting the references tab In the Customer tab: 5. Create an “Add new customer” button that pulls up a modal 6. Create the “Add new customer” modal. This modal prompts the user to fill out text fields that includes: Customer group, company name, company address, related documents, related notes, references 7. Add new references to the reference database. When adding references to a customer, include the following information about each reference: first name and last name, email address, phone number, reference type 8. Create a “Confirm” button that enters the new customer and references into their respective databases. 9. Write and administer appropriate tests on user verification 10. Write and administer appropriate tests for the Customer database 11. Write and administer appropriate tests for the References database 12. Write and administer appropriate tests for adding a new customer to the database

Dependencies
The user system must be created so that only administrators and sales owners can access this.

- **A.5.2 Removing a customer**

Task breakdown
1. Add a “Remove” button to the customer tab that appears when a customer is selected. When clicked, a modal appears confirming that the customer should be deleted. 2. Write code to delete customer from the customer database 3. Write and administer appropriate tests
Dependencies
A.5.1: - Customer tab is created - Customer database is created The user system must be created so that only administrators can access this.

- **A5.3 Edit a customer**

Task breakdown
1. Add the user type check functionality to confirm user is admin/sales owner 2. An edit customer popup needs to be made which will contain the information on customers for user to overwrite, as well as a “Confirm changes” button 3. Add an edit button to the customer tab when a customer is selected which will generate a popup 4. A function needs to be programmed to use newly written data to update user profile 5. Create and execute appropriate test to check whether new information is displayed under the edited user
Dependencies
User types sales owner and administrator exist and have assigned privileges User login page exists so user can be logged in and there is a function to check user type A5.1: Customer tab is created; Customer database is created and populated

- **A5.4 Search for query in customers**

Task breakdown
1. “Search” text field needs to be added 2. Searching function needs to be programmed to match input text to existing customer properties and added to Search text field 3. Create and administer appropriate test to check functionality
Dependencies

User login page exists so user can be logged in A5.1: Customer tab is created; Customer database is created and populated

- **A6.1 Adding a stage to the pipeline**

Task breakdown
1. Add stage popup needs to be made with a text field for Pipeline stage description and a text field "multiplier" for revenue multiplier which will also contain a Confirm button 2. "Add stage" button needs to be added in the admin module in the pipeline tab which triggers a popup 3. Create function to add the new stage in pipeline based on given information and add it to the Confirm button 4. Create and execute test to check whether new stage was created
Dependencies
A.2.1: Basic pipeline functions need to be created

- **A6.2 Removing a stage from the pipeline**

Task breakdown
1. Create popup to confirm remove stage containing a "Confirm" button 2. Add "Remove stage" button in admin module in the pipeline tab which triggers a popup 3. Create function to remove a stage in pipeline based on given information and add it to Confirm button 4. Test the removal function on previously made test cases under 6.1
Dependencies
A.2.1: Basic pipeline functions need to be created A 6.1: For testing purposes we should use the created test cases

- **A6.3 Change pipeline stage projected revenue multiplier**

Task breakdown
1. Add "Edit pipeline stage" popup which would have textboxes from use case 6.1 and "Save changes" button 2. Add "Edit" button in admin module in the pipeline tab which triggers a popup 3. The function to amend projected revenue multiplier of a stage in pipeline based on given information must be made and assigned to Save changes button 4. Test whether changes have been made and saved
Dependencies
A.2.1: Basic pipeline functions need to be created A 6.1: Multiplier field needs to be created

Project Plan

We split our project into 2-week sprints. We placed all the use cases that have a lot of dependents into sprint 1. This includes use case A1.1, A2.1 and others. Many of the use cases later are dependent on the use cases in sprint 1, therefore it is important to get these cases completed as soon as possible. Since Use cases 3.1 and 5.1 are dependent on use cases 1.1 and 2.1, we put them after the first two use cases in the sprint. In the next sprint, we placed use cases that were dependent on the cases in sprint 1 and followed a similar structure for the other sprints. This way, we can complete use cases without any delays in waiting for other use cases to be completed. This saves time and increases productivity by reducing the waiting time and dependencies for other use cases to be complete.

Furthermore, we prioritized "Must Have" use cases by making sure all must have cases are before the "Should Have" cases. This increases productivity by minimizing the time required to get to a working product. Finally, we tried to work on each section of use cases (e.g. A1 User Management) concurrently to maximize the number of feedback loops for each section. By building in vertical slices, we maximize the number of feedback loops for each section and adhere as closely to customer expectations as possible.

Our assumption is that there will be 4 developers working on the use cases. As there is a 2-week sprint, this gives a total of 40 working days among us to complete the use cases and tasks. We assume that the developers will pick up the tasks accordingly depending on their bandwidth and competency within the area of the task. Where there are gaps in the sprints in terms of the days, this is to account for any holidays, meetings, or any extra bugs or issues that might come up.

Within the sprints, each table represents a use case, and the table has each task listed as well as an estimate in days. Within the explanation we have discussed the priority of the task, as well as an explanation as to why a task should take a certain amount of time. We also discussed any dependencies that were listed with other tasks and what other tasks would have to be completed in order to complete this task.

Sprint 1

The use cases in this sprint add up to total of 37.5 working days for an individual, or 10 working days for a team of 4 engineers.

[Must Have] Use case 1.1: Total of 10 working days

Task for use case 1.1	Estimate (in working days per person)	Explanation
Create Button to Navigate user to Account Page (1)	½ a day	A simple front-end functionality that links to the account page.
Create User Interface for Account Page (2)	2 days	Requires more planning in terms of wireframe, design etc. Once this has been created, we can then work on the button for the admin to click that navigates to the account page.

Create a button to add role for each account (3)	½ a day	Simple front-end functionality. Will be started once the Account page UI has been created.
Create class for Sales Owner object and assign attributes for the Sales owner Object Functionality (4)	2 days	Involves implementing methods of the sales owner and assigning inheritances to the account object. This then allows us to add the sales owner role to the account later.
Create Drop down menu that outputs the roles (5)	½ a day	Simple front-end functionality. This should be done after the roles e.g., the sales owner role has been created.
Create Button to confirm change and let administrator add role to the account. (6)	½ a day	This should be done after the roles have been output in the drop down so the user can select the role. Then the user can confirm using the button we have created.
Create User interface to output feedback to the user after confirmation of the change. (7)	2 days	The Confirmation button should be created for this task to function. There are also designs and wireframes to consider before implementation.
Create functionality to navigate to the account page upon adding the role to the account. (8)	1 day	The account page must have already been created as well as all the functionalities for the confirm button.
Create and Run unit tests on account information, checking that the Sales Owner Role has been Added (9)	1 a day	Simply testing that the sales owner has been added and different scenarios involved in the use case such as the confirmation, the UI etc.

[Must Have] Use Case 2.1 - Total 10 working days

Task for use case 2.1	Estimate	Explanation
Create database table for pipeline to store all the lead information in it and be able to add new entries for new leads. (1)	1 day	We would need to get the information on the leads to put onto the database as well create necessary database relations.
Create an object for the lead and all the lead details as attributes for the lead object (2)	½ a day	This is a high **priority task** as most of the tasks in this use case are dependent on the actual lead object being created. **This should**

		be the first task to be completed for this use case.
Create button to allow the Sales Owner/Administrator to select pipeline tab. (3)	½ a day	Simple front-end functionality. For this to be done the user interface must be created and linked to the button.
Create User interface for the pipeline tab and link it to the button so the user navigates to the pipeline tab once the button is pressed. (4)	2 days	Wireframes or design must be considered before implementation. Once this has been done the button can be created and you can link the interface to the button.
Create button to add new lead (5)	½ a day	The lead object would have been created by the time this task has begun.
Create user interface for the add new lead pop up and link to button so the pop up appears once the user clicks the button. (6)	1 day	Not as complex as previous user interfaces as it is a pop up. Should link to the button that was created as per the previous task.
Create form for user to fill in the lead details. The details of the form for the user to add regarding the leads can be found in the CRM requirements. (7)	½ a day	Once we have created the lead object and have the information about the leads we can do this. Task 2, 3, 4 and 5 should be done ideally before this.
Create button to submit form and confirm details that the user has filled in. (8)	½ a day	This task is dependent on the database to store the leads being completed in task 1. As when the user submits the form it will ideally add the entry onto the database.
Create validation for the form, so if admin/Sales Owner has missed certain details from the form it will prompt them to fill in those details and prevent submission. (9)	½ a day	Simple form handling functionality. The UI of the form with all the input fields should be created first.
Create functionality to add the lead into the database table for the pipeline. Assign this method to the Sales Owner/Administrator object. (10)	1 day	This requires the sales Owner from the previous use case (use case 1.1 task 4) so that task should be done before this. The form should also be fully functional (task 7-9) before doing these tasks.
Create database table to store all events then log this event onto the database once the lead has been added. (11)	1 day	This can be done after all the other tasks above have been completed as we would have to save all the user activities as events then create the table to store these events.

Link account page to the submit form button. Create logic so upon confirmation, once the lead has been added and everything is correct, it will redirect to the accounts page. (12)	½ a day	The form and submit confirmation button must be complete and functional (tasks 7-10) before linking the page to the button.
Create and Run unit test to check that the lead has been added onto the database (13)	½ a day	Can be done before any of the tasks (in a test driven development format) depending on developer bandwidth, or after completion.

[Must Have] Use Case A3.1 - 8.5 person-days

Task for use case A3.1	Estimate	Explanation
Create sales invoice class (1)	½ day	Class to contain info about invoices.
Create invoices DB (2)	1 day	Create schema and import data. It would take a lot more than a day if manual moving was needed.
Check that user class can have sales owner/administrator roles (3)	½ day	Account functionality needed for verifying user
Link “generate invoice code” to verification that user is a sales owner/administrator and is logged in (4)	½ day	Quick check in case user somehow skipped the login screen or their session expired
Link code to verification that user is in the SalesRock module (5)	½ day	Verify user is in module just in case they accidentally access toggle view elsewhere
Add button “Generate invoice” that appears when lead is selected (6)	½ day	Depends on pipeline UI being created
Design “new invoice” popup with relevant fields to fill in and a “Generate invoice” button (7)	½ day	Formatting and style choices. Decide what inputs are allowed in each field. Separation of design and implementation lets people specialize into tasks they’re good at
Write code to generate popup upon button click (8)	½ day	Depends on button creation and popup code creation. Validate inputs for fields where they’re restricted.
Add code to take popup fields and generate an invoice (9)	1 day	Depends on DB, invoice class, and popup creation
Add code to send invoice to administrator system (10)	½ day	Depends on invoice class and DB creation
Design user feedback, such as a “successfully generated” popup (11)	½ day	Separation of design and implementation lets people specialize into tasks they’re good at
Write code to display feedback and close “new invoice” popup upon successful generation (12)	½ day	Depends on style choices in previous task
Write and run tests for popups and buttons (13)	½ day	Testing depends on code

Write and run tests for invoice generation and sending to administrator system (14)	1 day	Testing needs code and systems to already be implemented

[Must Have] Use Case A5.1 - 9 person-days

Task for use case A5.1	Estimate	Explanation
Check sure user is logged into an active account that is either an administrator or sales owner (1)	½ day	Account functionality needed for verifying user.
Create customers tab and a references tab in the SalesRock module (2)	½ day	Depends on general SalesRock UI
Create a Customer database and display entries upon selecting the customers tab (3)	1 day	Create schema and import data. Increased time if we must generate test data to populate the database with
Create a References database and display entries upon selecting the references tab. (4)	1 day	Create schema and import data. Increased time if we must generate test data to populate the database with
In the Customer tab, create an "Add new customer" button that pulls up a modal. Create the "Add new customer" modal. This modal prompts the user to fill out text fields that includes - Customer group - Company name - Company address - Related documents - Related notes - References (5)	1 day	Requires an existing customers database for the information to be added to
Add new references to the reference database. When adding references to a customer, include the following information about each reference - First name and last name - Email address - Phone number - Reference type (6)	1 day	Requires an existing references database for the information to be added to
Create a "Confirm" button that enters the new customer and references into their respective databases. (7)	1 ½ day	Dependent on the previous two tasks to validate the information. This step

		is communicating the results of validation with the user
Write and administer appropriate tests on user verification. (8)	½ day	Testing depends on code and systems to be implemented
Write and administer appropriate tests for the Customer database. (9)	1 day	Testing depends on database being created
Write and administer appropriate tests for the References database. (10)	1 day	Testing depends on database being created

Sprint 2

The use cases in this sprint add up to a total of 40 working days for an individual, or 8 working days for a team of 4 engineers.

[Must Have] Use case 1.2- Total 4 working days

Tasks for Use case 1.2	Estimate	Explanation
Create logic to check if the account already has a role. (1)	½ a day	This task can be completed once the tasks for the use case 1.1 are complete as by the functionality of adding a role would have been done.
Create Button to delete role on the account if the account has a role (2)	½ a day	Simple front end functionality. As mentioned above the tasks for use case 1.1 should be complete before this as well as the creation of the UI for the account page, as this is where the button will be placed.
Create user interface for the remove role pop up and create output which will prompt the user to confirm the change they have just made. (3)	1 day	This should be done once task 1 and 2 are done. Consider relevant designs and wireframes before starting.
Create Functionality to delete Sales Owner Attribute from the account upon confirmation from the user. (4)	1 day	**High priority task.** Can be done as separate functionality on the backend or done once task 1-3 are completed.
Create functionality to navigate to the account page upon deleting the Sales Owner role from the account. (5)	½ a day	Requires the Account page to be created as per Use case 1.1 task 2 as well as the delete functionality (task 1-4).

Create and run test to check that the Sales Owner Role has been removed. (6)	½ a day	Can be done before any of the tasks (in a test driven development format) depending on developer bandwidth, or after completion of the previous 5 tasks.

[Must Have] Use case 2.2- Total 7 working days

Task for use case 2.2	Estimate	Explanation
Create button for the user to select the lead (1)	½ a day	Pipeline tab UI (task 4 use case 2.1) must be created as this is where the button will be as well as the object for the lead (task 2 use case 2.1). Once these 2 are done then the button can be created which is a simple front end functionality.
Create button to edit lead information (2)	½ a day	Previous task must be created so the user can select the lead to edit.
Create UI for edit lead pop-up (3)	1 day	**High priority task**. As this is where most of the editing functionality will be taking place. Relevant designs and wireframes must be considered before implementation.
Create an input form for the user to input new values for the information that they want to edit. (4)	1 day	Implemented in the pop up so task 3 should be completed before beginning this task.
Create 'Save changes button' to confirm changes (5)	½ a day	This will be implemented within the form so the form and the UI (task 3 and 4) should be done before this task starts.
Create functionality to edit values of the lead upon confirmation. Assign this method to sales Owner/administrator object. (6)	2 days	**High Priority Task.** This can be done whilst the UI, the forms and the pop ups are completed concurrently and once the they are created link the back end (this task) to the front end (the UI) through the buttons created.
Create event for the editing of the lead info and log onto system events DB. (7)	½ a day	Requires the events database to be created first (task 11 use case 2.1) before completion.
Link pipeline tab to save changes button so user navigates to the pipeline tab once the button has been clicked. (8)	½ a day	Requires the UI for the pipeline tab to be created as well as tasks 1-6 to be completed for the link to be created.
Create and run unit tests to make sure that the new values are up to date with what the user has edited. (9)	½ a day	Can be done before any of the tasks (in a test driven development format) depending on developer bandwidth, or after completion of the previous 5 tasks.

[Must Have] Use case 2.3 - 8 working days

Task for use case A2.3	Estimate	Explanation
Create attributes of the different pipeline stages which will be stored in the lead object created. (1)	1 day	Different stages that the lead goes will have to be agreed and confirmed. The lead object will have to be created (task 2)
Create UI that outputs the stages that the lead is currently on within the pipeline tab (2)	1 day	**High priority task.** Previous task must be done so we have the different possible stages of the lead stored. Without this we won't know what stage the lead is on and won't be able to edit accordingly.so this should be done before the other tasks below. Pipeline tab should also be created before starting this task (use case 2.1 task 4)
Create a button on the lead that the user can click and drag.(3)	½ a day	Lead object should have been created to be able to link to the button and allow user to drag accordingly.
Create functionality to change the positioning of the lead once it has been dragged. Assign this method only for sales Owners/Administrator Object (4)	1 day	Sales owner object must be created (as per use case 1.1) as well tasks 1-3 so the user can drag the lead to a different stage.
Create functionality to update pipeline stage lead based on where user drags the lead to. This method is only assigned to Sales Owner/Administrator Object (5)	2 days	**High priority task.** Once task 1-4 is completed start this task. This is because the button is required as well as the dragging functionality in order for the functionality to work. Sales Owner object should also be created as that is the user type that has this right.
Create an event showing that the pipeline stage of the lead has been updated (6)	½ day	For this task the database of the events should have been created (Use case A2.1 task 11).
Log this event onto the events database by adding a new entry (7)	½ day	Previous task should have been completed in order to log the event.
Create and run tests to make sure that the current pipeline stage is up to date with what the Sales Owner/Administrator has edited. (8)	½ a day	Can be done before any of the tasks (in a test-driven development format) depending on developer bandwidth, or after completion of the previous 8 tasks.

[Must Have] Use Case A2.5 - 6.5 person-days

Task for use case A2.5	Estimate	Explanation

Check that user class can have active account (1)	½ day	Active account functionality needed for verifying user
Link toggle view code to verification of user having active account (2)	½ day	Quick check in case user somehow skipped the login screen or their session expired
Write code to verify the user being in the pipeline tab and link to toggle view code (3)	½ day	Verify user is in pipeline tab just in case they accidentally access toggle view elsewhere
Add a "Toggle view mode" button to the pipeline tab (4)	½ day	Depends on pipeline UI creation
Design UI for each view mode (5)	1 day	Formatting choices for headings, data, etc. Separation of design and implementation lets people specialize into tasks they're good at
Add code to turn data into UI for each view mode (6)	2 days	UI needs to have been designed in previous task before coding
Add code for button to control which visualization code is used (7)	½ day	UI code needs to have been written before button can switch between styles
Write and run tests on visualization code (8)	1 day	Code needs to be written before testing can happen, but tests can be written before code

[Must Have] Use Case A4.2 - 6.5 person-days

Task for use case A4.2	Estimate	Explanation
Check user is logged into an active account that is either an administrator, sales owner, or employee (1)	½ day	Account functionality needed for verifying user
Create visualization tab in the SalesRock module (2)	½ day	Depends on general SalesRock UI
Create KPI profiles database. Include an internal dependency of unique name. (3)	1 day	Create schema and import data. Increased time if we must generate test data to populate the database with
Create initial prompt to view profiles and build out the option to view predefined KPI profiles. Write code to fetch these entries from the database. (4)	1 day	Time required to layout the initial prompt and connect this with actual data from the new database
Create a KPI visualization. These 5 visualizations are stored in the predefined KPI profile database. (5)	1 ½ day	Dependent on the KPI profile database being created

Display the visualizations in accordance with the KPI profile. This means designing the page such that all aspects of the visualization are well presented for the user. (6)	1 day	This page depends on the visualization tab being created and the various visualizations from
Write and administer appropriate tests on user verification. (7)	½ day	Testing depends on code and systems to be implemented
Write and administer appropriate tests on the KPI profile database. (8)	1 day	Testing depends on database being created

[Must Have] Use Case A3.3 - 8 person-days

Task for use case A3.3	Estimate	Explanation
Check that user class can have sales owner/administrator role and can be active (1)	½ day	Account functionality needed for verifying user
Link code to verification that user is a sales owner/administrator, is logged in, and has an active account (2)	1 day	Quick check in case user somehow skipped the login screen or their session expired
Link code to verification that user is in SalesRock module (3)	½ day	Verify user is in module just in case they accidentally access toggle view elsewhere
Add “download” button to invoice (4)	½ day	Depends on invoice UI creation
Design PDF version of invoice (5)	1 day	Separation of design and implementation lets people specialize into tasks they’re good at
Add code to turn the information in invoice class into PDF (6)	1 day	Depends on invoice class creation and previous task
Add code to prompt browser to download invoice as PDF (7)	1 day	Depends on invoice UI and download button creation
Design user feedback on download, such as a notification (8)	½ day	Depends on invoice UI creation. Separation of design and implementation lets people specialize into tasks they’re good at
Add code to provide feedback on download (9)	½ day	Depends on choices in previous task
Write and run tests on user verification (10)	½ day	Testing needs code and systems to already be implemented, but tests can be written before code
Write and run tests on downloading (11)	1 day	Testing needs code and systems to already be implemented, but tests can be written before code

Sprint 3

This use cases in this sprint add up to a total of 39 working days for an individual or about 10 days for a team of 4 engineers.

[Must Have] Use case A2.4- Total of 2 working days

Task for Use case A2.4	Estimate	Explanation
Create a view edit history button that is outputted once the user clicks the "Select lead" button. (1)	½ a day	The select lead button should have already been done (A2.2 task 1) before this button is created.
Create User Interface for edit history pop-up that is outputted once the user clicks on the view edit history button (2)	1 day	**High Priority task.** We need this to be done so that the user can view the edit history and therefore satisfy the use case. The previous task would have needed to be completed so the UI can link to the button.
Create and run test to check that the edit history has been outputted and has been viewed. (3)	½ a day	Can be done before any of the tasks (in a test driven development format) depending on developer bandwidth, or after completion of the previous 3 tasks.

[Must Have] Use Case A3.2 - 11 person-days

Task for use case A3.2	Estimate	Explanation
Check that user class can have administrator role (1)	½ day	Account functionality needed for verifying user
Link code to verification that user is an administrator and logged in (2)	1 day	Quick check in case user somehow skipped the login screen or their session expired
Link code to verification that user is in SalesRock module (3)	½ day	Verify user is in module just in case they accidentally access toggle view elsewhere
Design error handling in the case of nonexisting invoice (4)	½ day	Separation of design and implementation lets people specialize into tasks they're good at
Add code to verify that invoice already exists, with error handling (5)	½ day	Depends on invoice DB creation. Check in case a missing/deleted invoice is accidentally edited
Add an "invoices" tab to SalesRock UI (6)	½ day	Depends on general SalesRock UI
Design invoices tab UI (7)	1 day	Separation of design and implementation lets people specialize into tasks they're good at

Add code to create invoices table from invoices DB (8)	1 day	Depends on invoice DB creation
Add code to create invoices tab UI upon button click (9)	2 days	Depends on choices in task 7 and table creation in task 8
Add code to make invoice display “Status” dropdown menu on selection (10)	½ day	Depends on invoice tab creation in previous task
Add code to change invoice status in DB when a status on the menu is selected (11)	½ day	Depends on invoice DB creation
Add code to update UI on DB changes (12)	1 day	Depends on invoice UI creation and invoice DB creation
Write and run tests on user verification (13)	½ day	Testing needs code and systems to already be implemented, but tests can be written before code
Write and run tests on status changing (14)	1 day	Testing needs code and systems to already be implemented, but tests can be written before code

[Must Have] Use Case A4.5 - 2.5 person-days

Task for use case A4.5	Estimate	Explanation
Check sure user is logged into an active account that is either an administrator, sales owner, or employee	½ day	Account functionality needed for verifying user
On the visualizations display, create a “Download as” button that pulls up a dropdown menu.	½ day	Requires an existing visualizations tab and visualization for A.4.2
Download the visualization in the selected format through the browser	½ day	Dependent on the previous task and on the visualization created in A.4.2
Write and administer appropriate tests on download format and ensure secure download	½ day	Testing depends on code and systems to be implemented
Write and administer appropriate tests on user verification	½ day	Testing depends on code and systems to be implemented

[Must Have] Use Case A5.3 - 3 person-days

Task for use case A5.3	Estimate	Explanation
Add the user type check functionality to confirm user is admin/sales owner	½ day	This is a function we presume would be created outside the use cases and is already implemented in other use

		cases, therefore should not take a long time to implement
An “Edit customer” popup needs to be made which will contain the information on customers for user to overwrite, as well as a “Confirm changes” button	½ day	A new popup window needs to be created for making changes to the customer information which would include the already filled information fields from 5.1 for user to edit before saving changes using Confirm changes button, which is why this use case is placed below it, however, due to the timings, it was set in Sprint 3
Add an edit button to the customer tab when a customer is selected which will generate a popup	½ day	A button must be added to the customer tab which would be a trigger for the popup
A function needs to be programmed to use newly written data to update user profile which will be assigned to Confirm changes button	1 day	A function to amend the customer information in the database must be created which would then be assigned to the Confirm changes button
Create and execute appropriate test to check whether new information is displayed under the edited user	½ day	This will be done after implementing the above, to make sure the functions are working

[Should Have] Use Case A2.6 - 5 person-days

Task for use case A2.6	Estimate	Explanation
Check that user class can have active account (1)	½ day	Active account functionality needed for verifying user
Link search code to verification of user having active account (2)	½ day	Quick check in case user somehow skipped the login screen or their session expired
Write code to verify the user being in the SalesRock Module and link to search code (3)	½ day	Verify user is in module just in case they accidentally access toggle view elsewhere
Add a “search” text field to the pipeline UI (4)	½ day	Dependent on pipeline UI creation
Write code to query pipeline DB using the text in the search field (5)	1 day	Requires pipeline DB to already be created
Link queried data to pipeline visualization (6)	½ day	Requires pipeline tab to be able to visualize lead data
Write and run tests on user verification (7)	½ day	Code needs to be written before testing can happen, but tests can be written before code
Write and run tests on pipeline redirection, DB querying, and data visualization (8)	1 day	Code needs to be written before testing can happen, but tests can be written before code

[Should Have] Use Case A4.1 - 10.5 person-days

Task for use case A4.1	Estimate	Explanation
Check that user class can be active (1)	½ day	Account functionality needed for verifying user
Link code to verification that user is logged in, on SalesRock visualization tab, and has an active account (2)	1 day	Quick check in case user somehow skipped the login screen or their session expired
Add code to check that there is an active visualization (3)	½ day	Verify just in case user somehow accesses a deleted visualization
Add “Change filter” button to visualization tab (4)	½ day	Depends on visualization UI creation
Design input fields for filters (5)	1 day	Decide which inputs are allowed, such as max input length, allowed characters, etc.
Design “change filter” popup with date range, stage, and project fee filters (6)	1 day	Formatting and style choices. Separation of design and implementation lets people specialize into tasks they’re good at
Add code to display popup upon button click (7)	1 day	Depends on choices in previous task, as well as button creation
Add code to validate filter inputs (8)	½ day	Depends on popup creation
Add “confirm change” button (9)	½ day	Depends on popup creation
Add code to set new filter based on inputs (10)	½ day	Depends on popup creation and visualization UI creation
Add code to query visualization DB with new filter (11)	1 day	Depends on visualization DB creation and previous task. Could probably reuse code from use case 4.1
Add code to reload visualization with new data (12)	1 day	Depends on data from previous task. Could probably reuse code from use case 4.1
Write and run tests on user verification (13)	½ day	Testing needs code and systems to already be implemented
Write and run tests on filter setting, DB querying, and visualization loading (14)	1 day	Testing needs code and systems to already be implemented

[Should Have] Use Case A4.3 - 5 person-days

Task for use case A4.3	Estimate	Explanation
Check sure user is logged into an active account that is either an administrator or sales owner (1)	½ day	Account functionality needed for verifying user
In the visualizations tab, add an option to create a custom profile onto the initial prompt (2)	½ day	Requires an existing visualizations tab and initial prompt

Create a “configuration tool” to create a custom profile. This creates a new entry that should be added to the KPI profile database. The configuration tool will prompt the user to - Fill out a unique name - Add filters - Select different graphs - Select data for the graphs - Choose if the new profile should be added to the “custom profiles” set (3)	1 day	Requires an existing KPI profiles database for the information to be added to
Create a “Save” button that gives feedback on the profile (4)	1 day	Dependent on the previous task to validate the information. This step is communicating the results of validation with the user
Display the new KPI profile visualizations in the same way as a predefined KPI profile from use case A.4.2 (5)	½ day	Dependent on the successful creation of a custom profile from the previous task and should be compared to the default visualization for A.4.2
Write and administer appropriate tests on user verification (6)	½ day	Testing depends on code and systems to be implemented
Write and administer appropriate tests on adding a new entry to the KPI profile	1 day	Testing depends on code and systems to be implemented

Sprint 4

This use cases in this sprint add up a total of 16 working days for an individual or about 8 days for a team of 4 engineers. This is the last sprint, and all the use cases would have been completed by the end of this sprint.

[Should Have] Use Case A4.4 - 4 person-days

Task for use case A4.4	Estimate	Explanation
Check sure user is logged into an active account that is either an administrator or sales owner (1)	½ day	Account functionality needed for verifying user
In the visualizations tab, add an option to select a custom profile onto the initial prompt (2)	½ day	Requires an existing visualizations tab, initial prompt, and existing custom profiles from A.4.3

Create a modal that allows the user to select one of the existing custom profiles. (3)	1 day	Requires existing custom profiles created in A.4.3. There is a lot of data to display in this pop up
Display the new KPI profile visualizations in the same way as a predefined KPI profile from use case A.4.2. (4)	1 day	Confirm visualization is the same as displayed via the route explained in A.4.3
Write and administer appropriate tests on user verification. (5)	½ day	Testing depends on code and systems to be implemented
Write and administer appropriate tests on the various profile selection through visualization. (6)	½ day	Testing depends on code and systems to be implemented

[Should Have] Use Case A5.2 - 2.5 person-days

Task for use case A5.2	Estimate	Explanation
Check sure user is logged into an active account that is either an administrator (1)	½ day	Account functionality needed for verifying user
Add a “Remove” button to the customer tab that appears when a customer is selected. When clicked, a modal appears confirming that the customer should be deleted. (2)	½ day	Requires and existing customers tab from case A.5.1
Write code to delete customer from the customer database (3)	½ day	Requires an existing references database for the information to be deleted from and the information to delete from the previous task
Write and administer appropriate tests	1 day	Testing depends on code and systems to be implemented

[Should Have] Use Case A5.4 - 2 person-days

Task for use case A5.4	Estimate	Explanation
“Search” text field to be added to Customer tab	½ day	A text box with placeholder “Search” would be added to the Customer tab which would enable searching through Queries (5.1 would need to be done beforehand for this, as a table with customers would be needed). This and it being a “Should Have” case was the reason for the placement into the final sprint.

Searching function needs to be programmed to match input text to existing customer properties and added to Search text field	1 day	A function to search through the queries based on the input in the Search text field would need to be created and implemented into the search box
Create and administer appropriate test to check functionality	½ day	This would just be texting by using the Search box to make sure the appropriate queries are showing

[Should Have] Use Case A6.1 - 2.5 person-days

Task for use case A6.1	Estimate	Explanation
"Add stage" popup needs to be made with a text field for Pipeline stage description and a text field "multiplier" for revenue multiplier which will also contain a Confirm button to save input	½ day	This use case was put into Sprint 4 A new popup window needs to be created which will gather information input by admin and have a confirm button which will save information input
"Add stage" button needs to be added in the admin module in the pipeline tab which triggers a popup	½ day	A button to enable the popup to appear will need to be added into the admin module (as this is a module that can only be accessed by admin, there is no need for extra user validation) As this would need the pipeline tab to exist, it is dependent on A2.1, and as it is set under "Should have" priority, we have assigned it to Sprint 4
Create function to add the new stage in pipeline based on given information and add it to the Confirm button	1 day	This is a function that will create a new stage in the pipeline based on the information added in the popup which will be assigned to the created Confirm button
Create and execute test to check whether new stage was created	½ day	This will be done after all the above functionalities have been added to check whether they are working.

[Should Have] Use Case A6.2 - 2.5 person-days

Task for use case A6.2	Estimate	Explanation
Create popup to confirm remove stage containing a "Confirm" button	½ day	A new popup window needs to be created which will contain a confirmation button
Add "Remove stage" button in admin module in the pipeline tab which triggers a popup	½ day	A Remove stage button will need to be added in the admin module which

		would cause the above popup to appear (as this is a module that can only be accessed by admin, there is no need for extra user validation)
Create function to remove a stage in pipeline based on given information and add it to the Confirm button	1 day	This is a function that will remove a stage in the pipeline selected by user which will be assigned to this use case's Confirm button
Test the removal function on previously made test cases under 6.1	½ day	This will be done after all the above functionalities to check that they all work as expected, they will be tested on test case from 6.1 as this increases efficiency due to not having to create new stages just for testing, hence it was put below it.

[Should Have] Use Case A6.3 - 2.5 person-days

Task for use case A6.3	Estimate	Explanation
Add "Edit pipeline stage" popup which would have textboxes from use case 6.1 and "Save changes" button	½ day	A new popup window needs to be created for making changes to the pipeline stage (for efficiency the code for fields in 6.1 can be copied, hence could be dependent on 6.1), so we have placed it below it.
Add "Edit" button in admin module in the pipeline tab which triggers a popup	½ day	An Edit button will need to be added in the admin module which would cause the above popup to appear (as this is a module that can only be accessed by admin, there is no need for extra user validation)
The function to amend projected revenue multiplier of a stage in pipeline based on given information must be made and assigned to Save changes button	1 day	This is a function that will edit a stage in the pipeline selected by user which will be assigned to this use case's Save changes button
Test whether changes have been made and saved	½ day	This will be done after the above functions are implemented to test out whether the expected outcome is generated

Non-Functional Requirements

Platform and portability

According to requirements EVR004, EVR005, and EVR006, SalesRock will be available in English, and perhaps in Dutch and/or Italian. These requirements are somewhat measurable, but could be improved by stating a proficiency standard, such as “Available in [language] with 0 grammatical and spelling errors as verified by a native speaker”

According to requirements EVR007, SalesRock must be compatible with PostgreSQL. This is a measurable requirement because "compatible with” has a generally understood meaning of something along the lines of “CRM module can call database operations in PostgreSQL without errors.”

According to requirements EVR008, EVR009, EVR010, EVR011, and EVR012, SalesRock must support Chrome v74+ and Firefox v66+, and could support Safari v10.11+, Edge 40+, and/or mobile browsers. These requirements are measurable because “supports [browser]” is generally understood to mean “runs without errors on [browser].” However, the mobile requirement could be improved by specifying which mobile browsers on which mobile operating systems on phones with which screen sizes.

According to requirements EVR013, SalesRock dependencies can be updated to newer versions. This is not really measurable and is difficult to measure because it’s hard to predict what parts of browser APIs will no longer be supported in the future. An improvement would look something like *(Must Have) To improve long-term support, SalesRock will not use deprecated code or call API methods that are planned to be phased out.*

The URD document states that SalesRock will be an addition to MyRock and will use the server and DB system provided by TU/e.

Response Time / Speed

As we mentioned in the previous assignment, the PFR requirements are measurable and tangible but could be improved by explicitly defining a suite of operations, browsers, operating systems, and internet connections to test response and feedback times with. Or a more restricted requirement could be used, such as:

(Must Have) The SalesRock module must create an invoice, a new lead, or a new visualization on Chrome version 74 or higher with an internet speed of at least 10Mb/s and MacOS 10.0 or later within 5 seconds.

They could also expand the requirements to cover more aspects of the application, such as registration or downloading:

(Must Have) The SalesRock module must be able to register a new user on Chrome version 74 or higher with an internet speed of at least 10Mb/s and MacOS 10.0 or later within 5 seconds.

Efficiency

There are some requirements which over efficiency to a certain extend such as the performance requirements (3.2.1) These requirements discuss the speed at which it will process any of the actions completed by the user including the load, create, edit and remove actions. From this we can gauge the

efficiency of the system. These requirements can be measured and fulfilled by simply measuring the time the system takes to process the actions done by the user.

However, this is not enough to cover efficiency completely. There are still no requirements that discuss certain parts of efficiency such as how much the system can process within a specified time, throughput, and storage capacity. Below are some examples of requirements they should include within efficiency that are tangible.

(must have) The application can process up to 100,000 user actions at a given time.

(must have) The application should not exceed more than 20% of CPU consumption under peak levels of volume and concurrency.

(must have) The application should not take up more than 2% of the processers time.

These requirements cover efficiency as we are given an understanding of the resources the system will take up whilst completing its task. This is measurable as within a given time or action, we can look at how much CPU consumption or processor time it is taking. If it exceeds what is in the requirements, we know that the efficiency requirements have not been met and it is not efficient. We can measure these by looking into the hardware usage at a particular time and check how much of the resources are being used and compare it with what has been stated in the requirements.

Availability

Regarding Availability, we are not given an exact measurement as to how often the system should be available within a given time. We are given availability in terms of languages such as EVR004, EVR005 and EVR 006. But this is not enough within the context that we are looking for when it comes to this metric. I would add extra requirements that would cover availability that are tangible for e.g.

(Must have): CRM module should be available 99.9% of the time for all users within a given month, where 0.01% of the time is for any downtime and maintenance issues.

This would have been a tangible requirement as we can test whether this has been satisfied or not by simply measuring the percentage of time that it is unavailable and analyze it accordingly. It may not have been included because the assumption is it will be available 100% of the time, however I still believe this is a requirement that must be included, so the user is still aware of when the software is or isn't operating.

Robustness and Recovery

There is no mention of probability of failure or what should happen when a failure occurs. This is important because no application is perfect and human and machine errors should be accounted for to mitigate the damage caused. This NF requirement is slightly less important because SalesRock is built on top of the MyRock module as well as the underlying database system and server provided by TU/e, and most failure handling depends on the backend systems. Furthermore, they assumed perfect use in the requirements document, which means human error is not considered. Some tangible and measurable requirements could be:

(Must Have) SalesRock can restart after a failure in 10 seconds or less

(Must Have) Under perfect use, SalesRock encounters critical failures once every month of runtime.
(Must Have) SalesRock must fail without modifying the underlying DB system, and must output an error log detailing user and system actions from the previous 5 minutes to help revert corrupted data.

Maintainability, Reusability, and Extensibility

We will make sure to reuse code for sections that are very similar or that include the same functionalities (as outlined in the plan above), we would make sure to comment every function in the code or any loop to. This would ensure that the code is very easy to maintain, expand or reuse in any new functions that may be needed. The one way to measure this would be to count the number of times functions were used within other functions. We could also count any segments in code that were not commented and through it calculate and show the amount of commented code as a percentage.

Reliability

In the original document, no mean time to failure or rate of failure were mentioned. We can presume that under the testing and debugging we would make sure to remove any bugs or warnings to ensure maximum reliability of the program. However, as external servers would be used, we can include the servers expected meantime to failure or its rate of failure as our won as our own, otherwise, we could include the times from our test to calculate the rate of failure in case there are any issues that could not be resolved as a measure.

More is also explained under Robustness and Recovery, as well as Availability.

Process

The original document failed to indicate any information regarding process (including the development methodology and cost and delivery date). As for the development process, this document outlines an agile methodology with a total of four sprints. These sprints are budgeted to be 2 weeks long each and are for a team of 4 engineers, with the exception of the final sprint, which is one week long. Each week has about 40 billable hours, so there are 133 total billable hours during the 7 weeklong project.

Ease of Use/Usability

The original document identifies which language the application must be available in and two optional languages. In this way, there are some tangible and measurable requirements. However, Ease of Use and Usability also expands to cover requirements covering training time and the number of help frames.

We propose the following chart to describe training time requirements. These requirements have been set to reflect the extent to which each role interacts with the application:

	Administrator	Sales Owner	Employee
Basic understanding	3 hours	2 hours	1 hour
Moderate understanding	5 hours	4 hours	2 hours
Advanced understanding	10 hours	8 hours	3 hours

Regarding help frames, there should be a “Help” button on the home page and on each additional tab (including the pipeline tab, invoices tab, visualization tab, customers tab, references tab) for a total of 6 instances to include help frames.

** Completed as a group of 4 developers. My contributions include: Task breakdown, dependencies and estimates for Use Cases: A1.1, A1.2, A2.1, A2.2, A2.3, A2.4. Nonfunctional requirements discussion: Efficiency and Availability.