Requirements Engineering

RepairSYS

Submitted By: Carraig Mc Nulty (T00196108)

Computing with Software Development

Date Submitted: 07/12/2018

**Table of Contents**

[1. Introduction/overview 4](#_Toc531971277)

[2. Functional Components 5](#_Toc531971278)

[3. User Requirements 6](#_Toc531971279)

[3.1. RepairSYS will manage Parts 6](#_Toc531971280)

[3.2. RepairSYS will process repairs 6](#_Toc531971281)

[3.3. RepairSYS will perform administrative reporting ` 6](#_Toc531971282)

[4. System Requirements 7](#_Toc531971283)

[4.1. System Level Use Case Diagram 7](#_Toc531971284)

[4.2. Manage Parts 8](#_Toc531971285)

[4.2.1. Add Part 8](#_Toc531971286)

[4.2.2. Update Part 10](#_Toc531971287)

[4.2.3. List Parts 12](#_Toc531971288)

[4.3. Manage Repairs 14](#_Toc531971289)

[4.3.1. Log Repair 14](#_Toc531971290)

[4.3.2. Estimate Cost 17](#_Toc531971291)

[4.3.3. Commence Repair 22](#_Toc531971292)

[4.3.4. Revise Repair 24](#_Toc531971293)

[4.3.5. Complete Repair 27](#_Toc531971294)

[4.3.6. List Repairs 30](#_Toc531971295)

[4.4. Admin 32](#_Toc531971296)

[4.4.1. Analyse Yearly Revenue 32](#_Toc531971297)

[4.4.2. Analyse Parts 34](#_Toc531971298)

[5. System Model 36](#_Toc531971299)

[5.1. Level-0 DFD 36](#_Toc531971300)

[5.2. Level-1 DFD 37](#_Toc531971301)

[5.3. Level-2 DFD (Process P1: Parts) 38](#_Toc531971302)

[5.4. Level-2 DFD (Process P2: Repairs) 39](#_Toc531971303)

[5.5. Level-2 DFD (Process P3: Admin) 40](#_Toc531971304)

[6. Data Model (Class Diagram) 41](#_Toc531971305)

[6.1. Class Diagram 41](#_Toc531971306)

[6.2. Relational Schema 42](#_Toc531971307)

[6.3. Database Schema 43](#_Toc531971308)

[7. Conclusion 46](#_Toc531971309)

[8. Appendices 47](#_Toc531971310)

[8.1. Appendix A – Sample List Parts 47](#_Toc531971311)

[8.2. Appendix B – Sample List Repairs 48](#_Toc531971312)

[8.3. Appendix C – Sample Graph: Revenue Analysis 48](#_Toc531971313)

[8.4. Appendix D – Sample Graph: Parts Analysis 49](#_Toc531971314)

# Introduction/overview

For my system I choose to do a computer repair system. My aim is to produce a system capable of managing the transaction processes within a repair shop. This includes management of the available parts for a repair, managing the different stages of a repair as it progresses form its beginning to its conclusion and finally tracking the administrative data that the business requires.

My primary user for this system would be employees or management involved in the computer repair business with no direct interaction required from the customer and so this specification was designed from a business perspective.

My intention is to include a Manage Parts module is to allow the user to add parts to the system by entering the part type, cost etc… Within this same module I intend to also include the possibility for a user to alter a part logged within the system. The final function of this module will be to simply list all the parts that are available to use for a repair.

For the second module I will provide capability for the user to Log a repair in the system once they have performed a diagnosis and determined whether the business is able to repair the laptop. I intend for this to be the main process of my system. Following this I will include the capability for the user to estimate the cost of a repair by adding parts to a basket while the system calculates the cost. Once complete this should email the customer and the cost of the repair and the user will not be able to alter the entry of the repair until the business has received a response from the customer. The last three processes in this will be allow the user to revise the parts added to a repair, commence the repair and complete the repairs, the purpose of these functions is to progress the repair from one stage to the next. The final process will filter the repairs in a list for the user.

For the third and final module of the system I will include administrator processes which will allow an admin user to access both a revenue analysis and a parts analysis which will provide graphs for the user.

# Functional Components

This section presets the functional components of the propsoed software system.

There are 3 main componets in the proposed system

# User Requirements

## RepairSYS will manage Parts

* + 1. RepairSYS will add a part.
    2. RepairSYS will update a part type
    3. RepairSYS will list stock parts.

## RepairSYS will process repairs

* + 1. RepairSYS will allow the user to log a repair.
    2. RepairSYS will estimate a repair cost.
    3. RepairSYS will allow the user to set a commencement status
    4. RepairSYS will allow the user to revise a logged repair
    5. RepairSYS will allow the user to set a completed status.
    6. RepairSYS will list repairs.

## RepairSYS will perform administrative reporting `

* + 1. RepairSYS will produce a yearly revenue analysis.
    2. RepairSYS will produce a parts analysis.

# System Requirements

RepairSYS has three main modules.

The Parts module will provide functions that will add a part, update a part type and list stock parts.

The Repair Module will allow a user to log a repair, produce an estimated repair cost, allow a user to set a commencement status within the repairs file, allow a user to set a completed status within the repairs file and will allow the user to review, list repairs and update a logged repair within the repairs file.

The Admin Module will allow a user to view a yearly revenue analysis and will produce a parts analysis.

## System Level Use Case Diagram

The following system level use case diagram illustrates the high-level system requirements.

Manager

Customer

Repair Technician

## Manage Parts

The Parts module will provide functions that will add a part, update a part type and list stock parts.

### Add Part

This function adds a part to the system each part is identified by a unique part id. A part can identified and defined by a Model Number.

<<Extends>>

<<Include>>

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Part** | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a part to the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Add Part function.  **Step 4:** The manager selects the part type from a dropdown list.  **Step 5:** The manager enters the part type details:   * ModelNo * Unit\_Price   **Step 7:** The manager confirms that the Part type is to be set. | **Step 2:** The system retrieves details of all part types from the **Part Types file**  **Step 3:** The system displays the UI.  **Step 6:** The system validates the data entered:   * All fields must be entered * Cost must be numeric and greater than zero * ModelNo must NOT already be registered   **Step 8:** The system assigns a unique identifier named Part\_ID.  **Step** **9**: The system saves the parts details in the **Parts File**   * Part\_ID * Part\_Model * Part\_Type * Unit\_Price   **Step 10**: The system displays a confirmation message.  **Step 11:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 5:** Invalid data is entered.    **Step 6:** The system displays an appropriate error message. |
| **Conclusions** | The new part is added to the Parts File. | |
| **Post conditions** | This part can now be used in repairs | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Part

This function updates the details of a selected part. For example, cost price,

<<Extends>>

<<Include>>

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Part** | |
| **Use Case Id** | 2 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow the user to alter a record in the parts file. | |
| **Preconditions** | A part must exist in the parts file to be updated. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the update part function.  **Step 3:** The manager chooses the part type to be updated from a dropdown list.  **Step 5:** The manager selects which part to update.  **Step 7:** The manager modifies the required details:   * Part\_Type * ModelNo * Unit\_Cost | **Step 2:** The system retrieves a list of registered part types from the parts file and displays in a dropdown list.  **Step 4:** The system retrieves all registered parts specified from the Part file and displays them on the UI.  **Step 6:** The System displays the details of the part chosen by the user and displays them on the UI  **Step 8:** The system validates data entered:   * Cost must be numeric and greater than zero * ModelNo must NOT already be registered   **Step 9:** The system updates the part details in the **Parts File.**   * Part\_Type * ModelNo * Unit\_Cost   **Step 8**: The system displays a confirmation message.  **Step 9:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 6:** Invalid data is entered.    **Step 7:** The system displays an appropriate error |
| **Conclusions** | The Part types details are updated in the parts file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### List Parts

This function will allow the user to view all parts in the parts file.

For Sample listing see Appendix A.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Parts** | |
| **Use Case Id** | 3 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow the user to view a list of stock parts in the parts file. | |
| **Preconditions** | A part must exist in the parts file listed. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the list parts function.  **Step 3:** The manager selects a part type to list | **Step 2:** The system retrieves the details of all parts from the parts file and fills a dropdown list.  **Step 4:** The system displays the details of all parts from the parts file that have been specified by the user. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Print option selected** | **Step 3:** Manger selects the print option | **Step 4:** System communicates with a printer to print the list.  **Step 5:** The system displays confirmation message |
| **Conclusions** | Parts and their details are retrieved from the parts file and displayed as a list. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Repairs

The Repair module will allow a user to log a repair, produce an estimated repair cost, allow a user to set a commencement status within the repairs file, allow a user to set a completed status within the repairs file, display logged repairs as a list and will allow the user to update a logged repair within the repairs file.

### Log Repair

This function will allow a user to log a repair each repair can be identified by a unique repair ID.

Repair Technician

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Log Repair** | |
| **Use Case Id** | 4 | |
| **Priority** | 1 | |
| **Source** | Repair Technician | |
| **Primary Business Actor** | Repair Technician | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow the user to log a repair to the repairs file. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The repair technician invokes the list repairs function.  **Step 4**: The technician enters relevant details that are pertinent to the stage of repair:   * Forename * Surname * Phone\_No * Email * Repair\_Description | **Step 2:** The system assigns a unique identifier named Repair\_ID.  **Step 3**: The system displays the UI  **Step 5:** The system validates the data entered:   * All fields must be entered * Cost must be numeric * Email must contain an ‘@’ symbol. * Phone\_No must be ten characters long   **Step 6:** When logging a repair, the Repair\_status will default to “Pending Repair”.  **Step 7:** The repair cost will be set to zero and the Log\_Date is set to the system date.  **Step 8:** The repair is saved in the **repairs file**.   * Repair\_ID * Forename * Surname * Phone\_No * Email * Repair\_Description * Repair\_Cost * Log\_Date * Status = ‘Pending’   **Step 9:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 7:** Invalid data is entered.    **Step 8:** The system displays an appropriate error message |
| **Conclusions** | The new repair is added to the Repair File. | |
| **Post conditions** | The Estimate cost function can be invoked on a repair with the status ‘Pending’ | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Estimate Cost

The function will allow a user to update a logged repair in the repairs file with an estimated cost.

Y

Error Message

|  |  |
| --- | --- |
| Repair Technician | System |
| Invoke Estimate Cost  Enter Qty  Select Part Type  Select Repair | Retrieve Repair Description & Email and Display on UI  Retrieve Part Types  Validate Qty  N  Valid?  Display UI  Retrieve Pending Repairs |
| Y  N  More Parts? | Reset UI  Display Confirmation Message  Send Email  Update Repair Cost  Save Repair Parts  Retrieve Email  Display Running cost and Parts on UI  Add Parts to basket  Update Running total |

Calculate Part Cost

Repair Technician

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Estimate Cost** | |
| **Use Case Id** | 5 | |
| **Priority** | 1 | |
| **Source** | Repair Technician | |
| **Primary Business Actor** | Repair Technician | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow the user to estimate the cost of a repair. | |
| **Preconditions** | A repair must be logged in the repairs file for an estimated cost to be added. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The repair technician invokes the Estimate Cost function.  **Step 4:** The technician selects the repair.  **Step 6:** The technician selects the Part\_Type to be used.  **Step 8:** The Technician selects a part from the list.  **Step 10**: The technician enters the desired quantity.  **Step 16:** The repair technician confirms required parts have been added. | **Step 2:** The system retrieves a summary of all the logged repairs with the status of ‘Pending’ from the repairs file and displays them on the UI.  **Step 5:** The system fills a combo box with registered parts from the parts file.  **Step 7:** The system displays a list of parts of this type.  **Step 11**: Thesystem validates the data entered:   * Quantity must be numeric and greater than 0 and less than 5   **Step 12:** The system calculates the cost of parts added to the repair (Quantity\*Part cost).  **Step 13:** The system updates the running total of the repair.  **Step 14:** The system adds parts to shopping basket  **Step 15:** If more parts are required go to **step 6.**  **Step 17**: Thesystem retrieves the email associated with the repair from the **repairs file** and notifies the customer.  **Step 19:** The system saves the parts used in the **Repair Parts file.**   * Repair\_Id * Part\_ID * Quantity   **Step 20:** The System updates the **repairs file.**   * Status = ‘Estimated’ * Repair\_Cost = Repair\_Cost + running total   **Step 21:** The system sends the customer an email asking them to get in contact with the store to confirm they want the user to go ahead with the repair.  **Step 22:** The system displays a confirmation message.  **Step 23:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 7:** Invalid data is entered.    **Step 8:** The system displays an appropriate error message |
| **Revised repair** | **Step 12:** The technician requires more parts | **Step 13:** The system displays a list of the part types and their details. |
| **Conclusions** | The repairs file is updated with a repair cost and a status of ‘Estimated’. The parts used in the repair is saved in the repair parts file. The customer is emailed | |
| **Post conditions** | The commence repair function can be invoked on a repair with the status ‘Estimated’. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Commence Repair

The function will allow a user to update a logged repair in the repairs file to the status ‘Processing’

Repair Technician

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Commence Repair** | |
| **Use Case Id** | 6 | |
| **Priority** | 1 | |
| **Source** | Repair Technician | |
| **Primary Business Actor** | Repair Technician | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow the user to change the status of a repair within the repairs file. | |
| **Preconditions** | A repair must have a status of ‘Estimated’ to be commenced.  A customer must respond to the user and agree to pay the full cost of the repair to before this function is invoked. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The repair technician invokes the commence repairs function.  **Step 4:** The technician selects which repair they would like to commence.  **Step 5:** The technician confirms that the repair is ready to commence.  **Step 8:** User confirms that the repair cost has been paid | **Step 2:** The system retrieves all the logged repairs with a status of ‘Estimated’ from the repairs file.  **Step 3:** System displays UI.  **Step 6:** The system retrieves the details from the **repairs file**  **Step 7:** The system asks the technician to confirm that the repair cost has been paid  **Step 9:** The system generates a Payment\_ Id for that repair’s payment.  **Step 10:** The system saves the details in the **Payments file**.   * Payment\_Id * Repair\_Id * Amount\_Paid * Pay\_Date = System date   **Step 11:** The system updates the **Repairs file** with a new status of ‘Processing’  **Step 12:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | A logged repair in the Repairs File will be updated with a status of ‘Processing’  The payments file is updated with the payment details. | |
| **Post conditions** | The Complete repair function can now be invoked on this repair. | |
| **Business Rules** | A customer must respond to the technician so that the repair can commence.  A repair can only be commenced if its status is ‘Estimated’ | |
| **Implementation Constraints** |  | |

### Revise Repair

This function will allow a user to revise the parts used in a logged repair from the repair parts file and allow them to add any additional parts required for the repair.

Repair Technician

<<Extends>>

<<includes>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Revise Repair** | |
| **Use Case Id** | 7 | |
| **Priority** | 1 | |
| **Source** | Repair Technician | |
| **Primary Business Actor** | Repair Technician | |
| **Other Participating Actors** | Manager | |
| **Description** | This function will allow a user to review the parts added to a logged repair | |
| **Preconditions** | This function can only be invoked on repairs with a status of ‘Estimated’ or ‘Processing’ | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The user invokes the Revise repair function.  **Step 3:** The user selects which repair they would like to revise.  **Step 5:** The technician confirms that a new part is to be added.  **Step 7:** The technician selects the Part\_Type to be used.  **Step 9:** The Technician selects a part from the list.  **Step 11**: The technician enters the desired quantity.  **Step 16:** The repair technician confirms required parts have been added.  **Step 18:** The repair technician confirms the details. | **Step 2:** The system retrieves all the logged repairs with a status of ‘Estimated’ or ‘Processing’ from the repairs file and displays on the UI.  **Step 4:** The system retrieves the part details from the repair parts file and calculates the current running cost of the repair.  **Step 6**: The system displays a list of the part types from the parts file.  **Step 8:** The system displays a list of parts of this type.  **Step 10:** The system asks the user to enter the quantity of the part specified.  **Step 12**: Thesystem validates the data entered:   * Quantity must be numeric and greater than 0   **Step 13:** The system calculates the cost of parts added to the repair (Quantity\*Part cost).  **Step 14:** The system updates the running total of repair.  **Step 15:** The system adds parts to shopping basket  **Step 17:** If more parts are required go to **step 6.**  **Step 19:** The system updates the repair in the repairs file.   * Repair\_Cost = running total * Status = ‘Estimated’.   **Step 20**: Thesystem retrieves the email associated with the repair from the **repairs file** and notifies the customer that their repair has been revised.  **Step 22:** The system saves new parts in the **Repair Parts file.**   * Repair\_Id * Part\_Id * Quantity   **Step 21:** The system sends the customer an email asking them to get in contact with the store to confirm they want the user to go ahead with the repair.  **Step 22:** The system displays a confirmation message.  **Step 23:** The system clears the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 6:** Invalid data is entered.    **Step 7:** The system displays an appropriate error message |
| **Conclusions** | Repair Parts are updated for a repair | |
| **Post conditions** | Repair Status is set ‘Estimated’ | |
| **Business Rules** |  | |
| **Implementation Constraints** | A repair cannot be revised if its status has been changed to ‘Completed’ | |

### Complete Repair

This function will allow the user to set the status of a repair to ‘Completed’.

Repair Technician

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Complete Repair** | |
| **Use Case Id** | 8 | |
| **Priority** | 1 | |
| **Source** | Repair Technician | |
| **Primary Business Actor** | Repair Technician | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow the user to change a repair status to completed. | |
| **Preconditions** | A repair must be of status ‘Processing’ within the repairs file to be completed. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the complete repair function.  **Step 4:** The technician choses a repair to complete.  **Step 6:** the technician confirms the repair is completed | **Step 2:** The system retrieves all repairs with a status of ‘Processing’.  **Step 3:** The system displays the UI.  **Step 5**: The system shows all of details of the selected repair from the **Repairs File**.  **Step 7:** The system updates the **repair file**   * Completed\_Date = System date * Status = ‘Completed’   **Step 8:** The **repairs file** is updated with the new status  **Step 9:** The system emails the customer.  **Step 10:** The system displays a confirmation message.  **Step 11:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Print option selected** | **Step 6:** Manger selects print option.  **Step 7:** the technician confirms the repair is completed | **Step 8:** The system updates the **repair file**   * Completed\_Date = System date * Status = ‘completed’   **Step 9:** The **repairs file** is updated with the new status  **Step 10:** The system emails the customer.  **Step 11:** The system prints invoice.  **Step 12:** The system displays a confirmation message.  **Step 13:** The system clears the UI |
| **Conclusions** | The repair’s status is updated to ‘Completed’ within the repairs file. The customer is emailed. | |
| **Post conditions** | The customer can now collect their repair | |
| **Business Rules** | A repair can only be completed if its status is ‘Processing’  Any unclaimed repair with a status of ‘Completed’ for more than six months is business property. | |
| **Implementation Constraints** |  | |

### List Repairs

This function will allow the user to view a list of logged repairs in the repairs file.

For sample listing see: Appendix B – Sample List Repairs

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Repairs** | |
| **Use Case Id** | 9 | |
| **Priority** | 3 | |
| **Source** | Repair Technician | |
| **Primary Business Actor** | Repair Technician | |
| **Other Participating Actors** | Manager | |
| **Description** | This function will allow the user to view a list of repairs logged within the repairs file. | |
| **Preconditions** | A repair must be logged within the repairs file to be listed. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the list repairs function. | **Step 2:** The system retrieves all repairs from the **repairs file**  **Step 3:** The system lists all repairs them on the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **List By Status** | **Step 3:** The technician selects the repair status they want to list by | **Step 2:** The system retrieves all repairs of the selected status from the **repairs file** displays the UI  **Step 3:** The system lists them on the UI. |
| **Print option selected** | **Step 3:** Manger selects the print option | **Step 4:**  System communicates with a printer to print the list.  **Step 5:** The system displays confirmation message |
| **Conclusions** | All registered repairs are retrieved from the repairs file and displayed as a list. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Admin

This will allow a user to view a yearly revenue analysis and will produce a parts analysis.

### Analyse Yearly Revenue

This function will allow a user to retrieve the revenue of a logged repair in the repairs file from a chosen year and display them on a graph.

For sample graph see: Appendix C – Sample Graph: Revenue Analysis

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Yearly Revenue** | |
| **Use Case Id** | 10 | |
| **Priority** | 3 | |
| **Source** | Manger | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow a user to review a yearly revenue analysis of a selected fiscal year up to the last completed month. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Analyse Yearly Revenue function.  **Step 3:** The manager selects which year they would like to Analyse. | **Step 2:** The system loads a combo box with years with a completed repair from the **Payments File**  **Step 4**: The system displays revenue of the chosen year in the form of a graph with which displays the monthly revenue of completed repairs for a selected year . |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Print option selected** | **Step 3:** Manger selects the print option | **Step 4:** System communicates with a printer to print the graph.  **Step 5:** The system displays confirmation message |
| **Conclusions** | The system displays a graph displaying the revenue from past years. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Analyse Parts

This function will allow a user to retrieve the revenue of Parts logged in the repair parts file from a chosen year and display them on a graph.

For sample graph see: Appendix D – Sample Graph: Parts Analysis

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Parts Used** | |
| **Use Case Id** | 11 | |
| **Priority** | 3 | |
| **Source** | Manger | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function will allow a user to review revenue analysis of a part types and individual parts up to the last completed month. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Analyse Parts function.  **Step 3:** The manager selects which year they would like to Analyse. | **Step 2:** The system asks the user what year they would like to examine.  **Step 4:** The system retrieves the 10 most used parts from the **Repair Parts File**  **Step 5**: The system shows the model number and the number of times this model was used for the top ten used parts in the selected year in the form of a graph. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Print option selected** | **Step 5:** Manger selects the print option | **Step 6**: The system shows the chosen year’s overall part revenue over the course of the fiscal year in the form of a graph.  **Step 7:** System prints graph  **Step 8:** The system displays confirmation message |
| **Conclusions** | The system displays a graph displaying the revenue from past years. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

RepairSYS

Customer

Repair Details

Invoice

## Level-1 DFD

Part Type Details

Customer Details

Customer

Parts Type File

D1

Repair Details

Manage Parts

P1

Process Repair

P2

Repair Part Details

Part Details

Parts File

D2

Repairs File

D3

Payment Details

Part Details

Payments File

D5

Repair Parts File

D4

Payment Details

Repair Part Details

Perform Admin

P3

Part Details

## Level-2 DFD (Process P1: Parts)

Part Details

Part Details

Parts File

D2

Part Details

Part Details

Add Part

P1.1

List Parts

P1.3

Update Parts

P1.2

Part Type Details

Part Types File

D1

Part Type Details

Part Type Details

## Level-2 DFD (Process P2: Repairs)

Part Details

Part Details

Part File

D2

Log Repair

P2.1

List Repair

P2.6

Estimate Cost

P2.2

Commence Repair

P2.3

Complete Repair

P2.5

Revise Repair

P2.4

Payments File

D5

Repairs File

D3

Repair Parts File

D4

Customer

Customer Details

Repair Details

Repair Details

Repair Status

Repair Details

Repair Details

Estimated Cost

Repair Parts Details

Repair Details

Repair Status

Repair Details

Repair Parts Details

Repair Parts Details

Repair Details

Payment Details

Part File

D2

Payment

Customer

## Level-2 DFD (Process P3: Admin)

Payment Amount

Repair\_ID

Repairs File

D2

Analyse Parts

Used

P3.2

Part Model

Parts File

D2

Repair Parts File

D2

Repair Parts Details

Payments File

D5

Analyse Yearly Revenue

P3.1

# Data Model (Class Diagram)

Brief introduction…

## Class Diagram

Part\_Type

* Part\_Type\*
* Description

Repair

* Repair\_ID\*
* Repair\_Cost
* Status
* Forename
* Surname
* Phone\_No
* Email
* Repair\_Description
* Log\_Date
* Completion\_Date

1

Has a

Payment

* Payment\_Id\*
* Amount\_Paid
* Pay\_Date

0.. \*

Has a

1

Part

* Part\_ID\*
* Part\_Model
* Unit\_Price

1

1

1

Has a

Repair Part

* Quantity
* Cost

1

0.. \*

Has a

## Relational Schema

**Part**\_**Types** (Part\_Type, Description)

**Parts** (Part\_ID, Part\_Model, Unit\_Price, Part\_Type)

**Repairs** (Repair\_ID, Repair\_Cost, Status, Forename, Surname, Completion\_Date,

**Repair**\_**Parts** (Repair\_Id, Part\_ID, Quantity, Unit\_Price,)

Log\_Date , Phone\_No, Repair\_Description)

**Payments** (Payment\_Id, Amount\_Paid, Pay\_Date, Repair\_Id)

## Database Schema

A definition of the database to be implemented.

**Schema**: RepairSYS

**Relation**: Part Types

Attributes:

Part\_Type varchar (30)

Description varhar(90)

**Primary Key:** Part\_Type

**Relation**: Parts

Attributes:

Part\_ID numeric (5)

Part\_Model varChar (35)

Part\_Type varChar (20)

Unit\_Price (6,2)

**Primary Key:** Part\_ID

**Foreign Key:** Part\_Type References Part\_Types

**Relation**: Repairs

Attributes:

Repair\_ID numeric (5)

Repair\_Cost numeric (6,2)

Status varhar (20) CHECK Status IN (‘Pending’, ‘Estimated’,‘Processing’, ’Completed’)

Forename varchar (25)

Surname char (25)

Phone\_No varchar (10)

Email varchar(80)

Repair\_Description (125)

Log\_Date Date

Completion\_Date date

**Primary Key:** Repair\_Id

**Relation**: Repair Parts

Attributes:

Repair\_Id numeric (5)

Part\_ID numeric (5)

Quantity numeric (1)

**Primary Key:** Payment\_Id

**Foreign Key:** Repair\_Id References Repairs

**Foreign Key:** Part\_ID References Parts

**Relation**: Payments

Attributes:

Payment\_Id numeric (5)

Repair\_Id numeric (5)

Amount\_Paid (6,2)

Pay\_Date Date

**Primary Key:** Payment\_Id

**Foreign Key:** Repair\_Id References Repairs

# Conclusion

I ultimately achieved what I had set out to do at the beginning of this specification though not in the same way that I had originally thought.

My parts module was completed as expected and works as I expected it to.

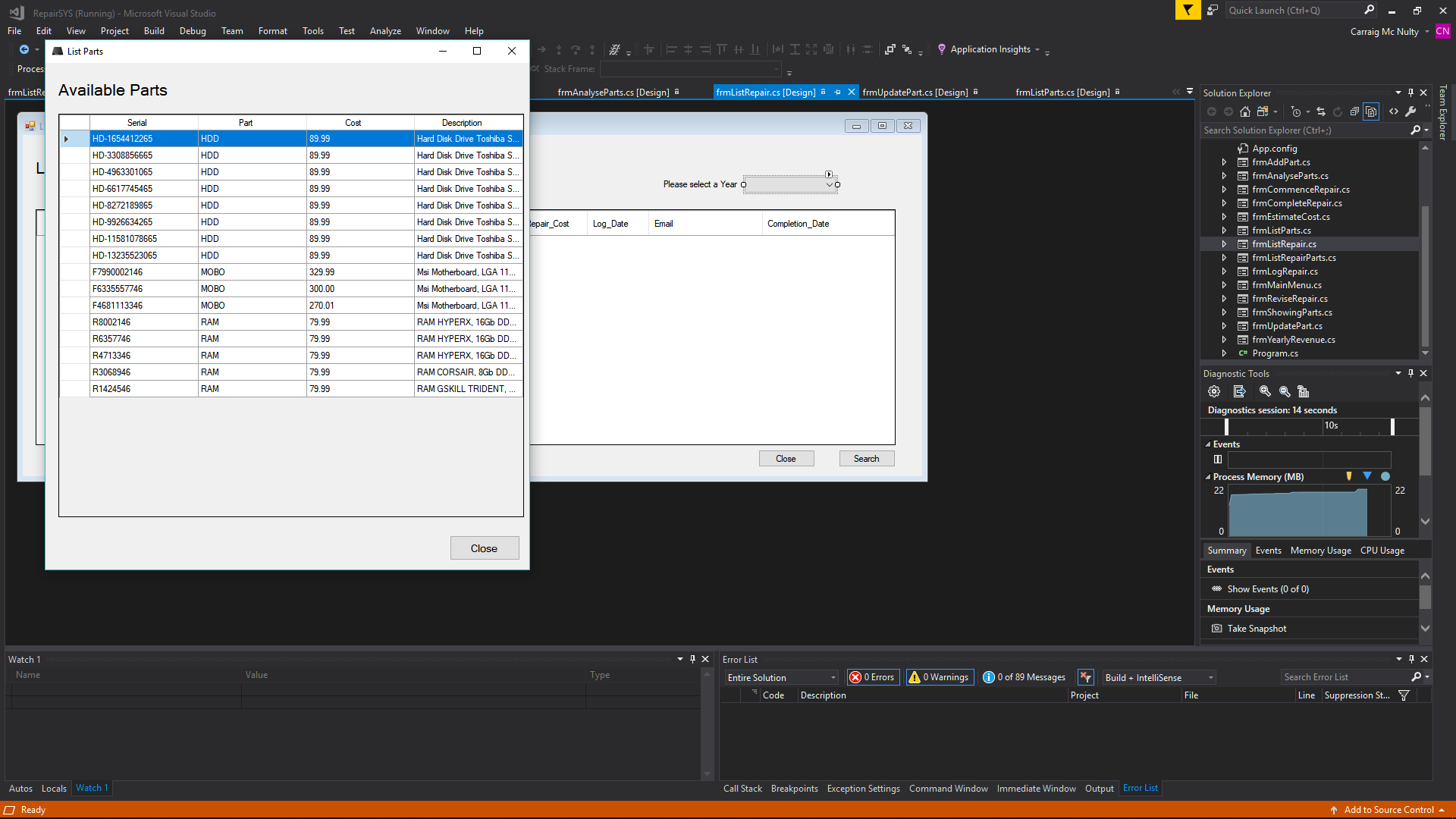
For my main module, Manage Repairs, Instead of Log Repair being my main process estimate repair becamethis and was much more complex from design perspective than I originally anticipated. Commence repair, complete repair and list repairs worked as expected allowing me to progress the repair from stage to the next and show the repairs that have been logged in the system. However, revise repair proved to be a difficult balancing act. This function was difficult to keep self-contained without it needing to add more to rely on other functions that will take my specification out of scope.

The final administrative part of the specification was handled and performed as expected. The user can view the revenue income on a graph and view parts sold on a graph.

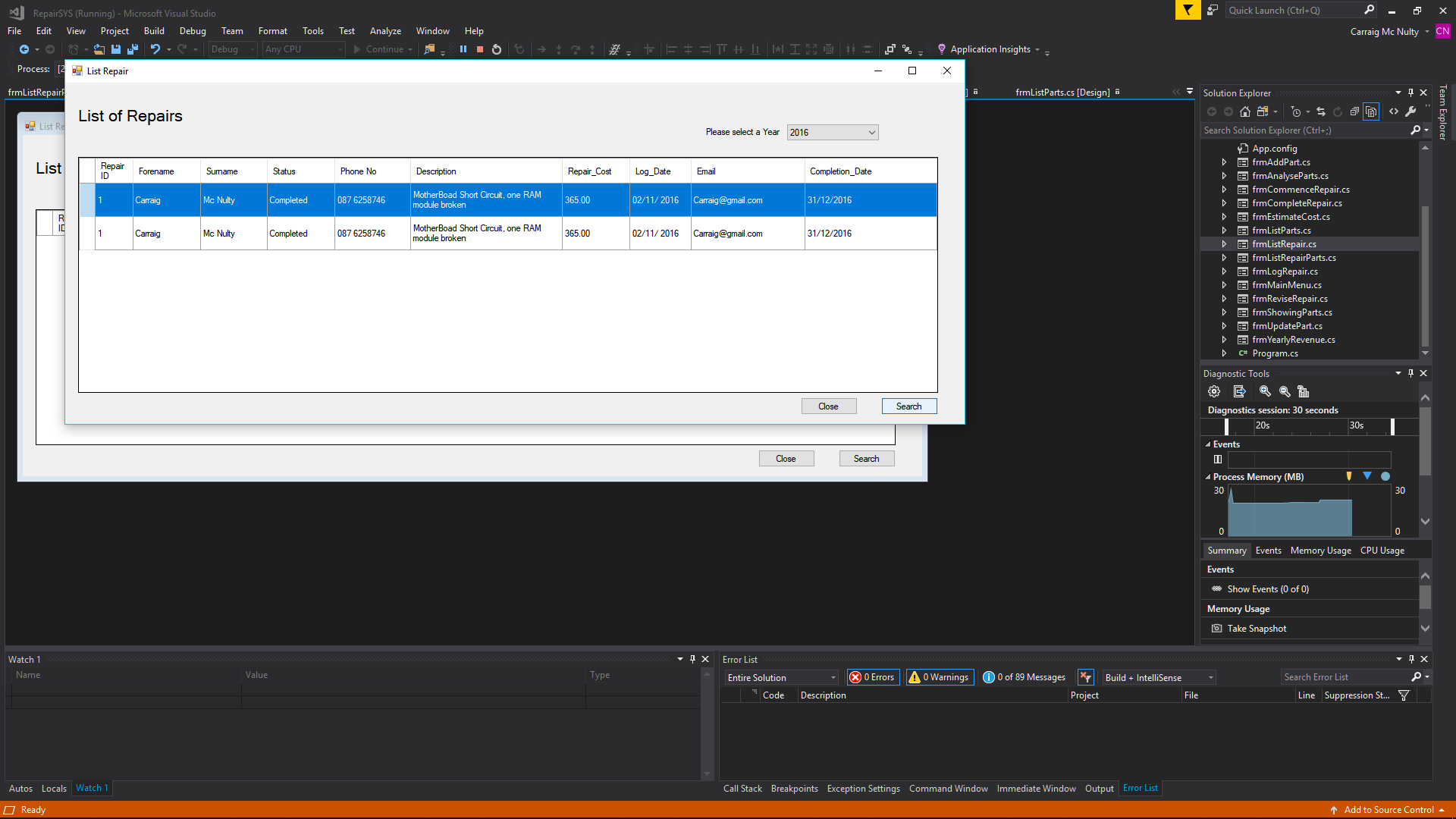
Something that I overlooked in the specification was the ability to track payments something which I implemented as part of the Estimate Cost process.

# Appendices

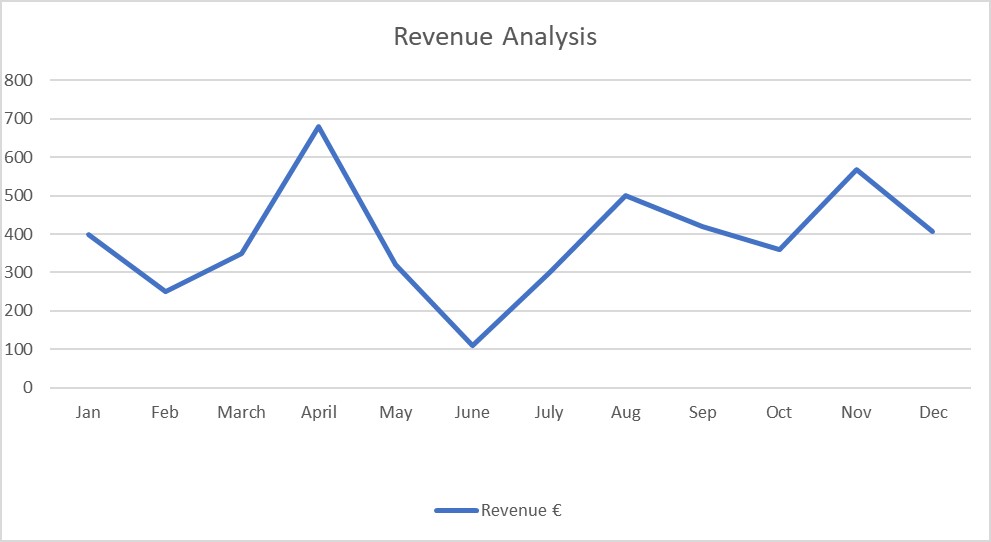
## Appendix A – Sample List Parts



## Appendix B – Sample List Repairs



## Appendix C – Sample Graph: Revenue Analysis



## Appendix D – Sample Graph: Parts Analysis

