**SalesRunner 2012 Iteration 3 Report**

**Group #1**

**Student Names: William Lancaster, Eliezer Mar Manarang, Wesley Henderson, Egor Gerasimenko, Mike Badmington**

**Due: 9/4/2012**

Table of Contents

[1.0 Iteration Overview 1](#_Toc321524339)

[2.0 System Requirements Change 1](#_Toc321524340)

[3.0 Overview of Work Completed 4](#_Toc321524341)

[3.1 Presentation Layer 4](#_Toc321524342)

[3.2 Application Layer 4](#_Toc321524343)

[3.3 Database Layer 4](#_Toc321524344)

[3.4 Release Task Completion Overview 4](#_Toc321524345)

[3.5 Source Control Server and Continuous Integration 7](#_Toc321524346)

[3.6 Source Code Analysis 7](#_Toc321524347)

[3.7 Software Tests 8](#_Toc321524348)

[3.8 Future Release Plans 8](#_Toc321524349)

[4.0 SalesRunner Updated Storyboard (Week 3) 9](#_Toc321524350)

[4.1 Verification 9](#_Toc321524351)

[4.2 Order Management 9](#_Toc321524352)

[4.3 Customer Management 10](#_Toc321524353)

[4.4 Salesmen Management 11](#_Toc321524354)

[4.5 Salesman Routing 12](#_Toc321524355)

[4.6 Extra Features 13](#_Toc321524356)

[4.7 New Features 13](#_Toc321524357)

[5.0 Detailed Member Efforts 14](#_Toc321524358)

[6.0 References 15](#_Toc321524359)

[Appendix A – Customer Critique 16](#_Toc321524360)

[Appendix B – Iteration 2 Retrospective 17](#_Toc321524361)

[Appendix C – Iteration 3 Meeting Details 19](#_Toc321524362)

[Appendix D – Storyboard Progress 24](#_Toc321524363)

# 1.0 Iteration Overview

Our goal for this iteration was to complete the implementation of the base features agreed upon with our customers concerning the SalesRunner 2012 application. We have succeeded in meeting these requirements, and in fact have successfully met a changed requirement specified in section 2 below as well.

While working on this project, we have thought of several improvements that could be made to the SalesRunner 2012 application for future releases. These details are specified in section 3 below. This document will first document the change in requirements that was undertaken during this iteration, and then proceed into an in-depth analysis of the work completed for this project. Finally, the snapshots of our storyboard throughout the three weeks of iteration 3 will show our progression of work (and will contain several new stories to meet changed requirements).

This document also contains meeting notes for this iteration, as well as notes on the retrospective meeting held regarding iteration 2.

# 2.0 System Requirements Change

There was a requirements change discussed with our customers regarding the functionality of the order division system. The initial plan was to have managers create an order and assign it to a specific salesman, or to automatically assign it to an available salesman and this salesman would then simply finish their assigned routes for the day. In this case, the routing algorithm would simply plot the most efficient route from order to order for a particular salesman.

However, our customers believed this would not allow an optimal amount of orders to be handled in a day. Instead, they proposed that when an order is created, it is simply added to a list of orders for a particular day, and any salesman may claim a particular order for that day. This allows salesman who wish to complete many orders in a day do so, while slower salesman may complete only a few. Considering a salesman’s pay is based upon completed orders, this could better drive the competition between salesmen to quickly (but successfully) fill orders.

The routing necessary for this change is rather simple. Instead of providing a salesman with a defined route for their orders, the salesman can instead choose to receive their most optimal order according to where they are. That is, a salesman can press a “Route” button on their orders page, and this will provide them with their best choice for an order. The salesman can then choose to “claim” this order, and thus it becomes unavailable for other salesman to choose, or the salesman may cancel and not receive an order. Thus, the salesman does not have the option of denying an order; if they do not wish to do a particular order, their only course of action is to wait until their optimal order is taken by another salesman. As such, this helps to ensure that **all** orders will get finished one way or another.

This requirement change affects several user stories:

|  |
| --- |
| A User Manually Assigns an Order to a Salesman |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Click on the “Orders” tab in the menu bar to show the summary of outstanding orders to be completed. Then edit and press the “Save” button to save the changes. |
| **Post Condition:** Saved changes will be registered in the database. Also, the Order and Salesman entity should be linked via a foreign key. Any changes to the foreign key (if the salesman is changed) must not violate any integrity of the database. |
| Priority: Medium |

With this new order routing requirement, a salesman is no longer manually assigned orders; a salesman must “assign” an order to themselves.

|  |
| --- |
| A User Automatically Assigns an Order to a Salesman |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Click on the “Orders” tab in the menu bar to show the summary of outstanding orders to be completed. Then edit the form (see Edit Form feature). Then on the “Assigned to” status, change the setting to be automatic. Then press the “Save” button to save the changes. |
| **Post Condition:** Saved changes must be registered in the database. The application layer for priority and routing distance calculation must assign the order to a specific salesman. Order entity and Salesman entity must be linked via foreign key. There should be no integrity violation in the database. |
| Priority: High |

The same goes for having an order automatically assigned to a salesman. With these system changes, these two user stories must be removed.

|  |
| --- |
| A User Views Their Order Routes |
| **Precondition:** Salesman user must be logged on to the system. |
| **Steps:** Click on the Orders tab to see the summary of outstanding orders that the salesman needs to fulfill. Then press the “Get Route” button on the top right corner of the summary form. The routing application layer will calculate the route for the next location order that needs to be done. |
| Post Condition: None |
| Priority: High |

Salesmen can no longer view orders assigned to them in this fashion. A salesman can only view one order; the order that they have claimed. Salesmen do not have the authority to view every order for a particular day, only their most optimal order at a given time, and only if they have claimed it.

This implies the necessity of several new user stories which we have discussed with our customer. They are as follows:

|  |
| --- |
| A User Assigns Themselves (Claims) an Order |
| **Precondition:** A salesman must be logged on to the system. |
| **Steps:** User navigates to the Routing tab. A pop-up appears, asking whether the source route should be the salesman’s home or from the last order they completed. If the user presses home, the routing algorithm uses the user’s home address, and if the user pressed Last Order Visit, the routing algorithm uses the last completed order’s visit address for the “From” field in routing.  The routing algorithm finds the most optimal order for that salesman to undertake (via the calculation lowest return value for (distance to destination) – (order priority level)), and gives the user the option to claim or cancel the order. If the salesman claims the order, they have assigned themselves to that order. Otherwise, the user is returned to the order page. |
| **Post Condition:** A salesman claims their chosen order and can now fulfill that order. |
| **Priority**: High |

This story handles the entire routing and claiming process for orders.

|  |
| --- |
| A User Views/Edits Their Claimed Order Details |
| **Precondition:** A salesman must be logged on to the system and has claimed an order. |
| **Steps:** User navigates to the order tab. The user’s previously claimed order details will appear, showing a Bing Maps route from their source location to the order’s visit location, as well as several fields which can be viewed/edited. These include adding items that were successfully sold to the customer (upon order completion), and changing the order’s State (Complete, Incomplete, or Delayed). The user solidifies the order when they set the state to complete; this updates the order completion date and all order settings become finalized. |
| **Post Condition:** A salesman has viewed/edited/completed their claimed order. |
| **Priority**: High |

This story handles the viewing/editing/completion of claimed orders.

The technical design changes we needed to make to accommodate this change in requirements were not vast. In fact, it simply amounted to removing the functionality it replaces, and creating a reasonable view with the necessary underlying business logic to do what it requires. That is, interfacing with Bing Maps to retrieve distances to all possible orders for a day, computing which route would be optimal on a singular per-salesman basis, and recommending that route to them. The claimed route page is no different than what we had before for that purpose.

# 3.0 Overview of Work Completed

This section details the functionality that has been implemented for the first release of the SalesRunner 2012 application. The functionality that has been completed past what was completed in iteration 1 and 2 is described in section 3.1, 3.2, and 3.3 according to the basic three layers of software architecture: presentation, application, and database layers. The presentation layer denotes what the user of the system actually sees. The application layer describes the underlying functionality; the computations which get work done. Finally, the database layer describes how the database was implemented.

A summarization table showing all that was planned and completed for this iteration will be shown in section 3.4. Next, details regarding the source control server and continuous integration tools we used for this project will be described in detail in section 3.5. After this, an analysis of the source code written for this project according to its software quality attributes will be given in section 3.6, followed by a synopsis of the testing coverage for the system in section 3.7. Finally, a brief overview of several features that could make their way into future SalesRunner releases are covered in section 3.8.

## 3.1 Presentation Layer

The new order and routing tabs for salesmen users were added, as well as the pop-up window mentioned in the “A User Assigns Themselves (Claims) an Order” user story. Small cosmetic changes were introduced throughout the GUI, notably having many of the options for orders removed in favour of simplicity and for following the change in requirements detailed in section 2. Finally, button styles were added as a settings option for users looking for a different aesthetic for their user account.

## 3.2 Application Layer

The routing functionality was added to facilitate the two new user stories, as well as the underlying logic connecting user accounts to their style settings they have chosen.

## 3.3 Database Layer

The “Item” entity in the database was more fully fleshed out to allow for salesmen to add sales items to a successful order, and the “Order” entity had several fields removed, leaving only an order number, state, salesman who claimed the order, and a connection to up to many sold items.

## 3.4 Release Task Completion Overview

This section contains a table labeling a particular iteration and its work velocity, the user stories planned for completion in that iteration, the planned tasks necessary for accomplishing those user stories, their priority, and finally whether or not those tasks were successfully implemented.

Note that if a planned task has been labeled as implemented in a particular planned story, it will not be repeated despite being a necessary for another planned task. It can simply be assumed that while that task was necessary, it was already taken care of, and thus is now not a “planned” task.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteration** | **Planned Stories** | **Planned Tasks** | **Priority** | **Implemented** |
| 1  Velocity = 40 | 1.1 “A User Logs In” | 1.1.1 Create the GUI login page | MUST | Yes |
|  |  | 1.1.2 Create a User table in the database | MUST | Yes |
|  |  | 1.1.3 Allow a successful login to a user’s page | MUST | Yes |
|  | 1.2 “A User Logs Out” | 1.2.1 Use the created material for story 1.1 and provide an invalid credential warning when the input fields do not match a database user | MUST | Yes |
|  | 1.3 “A User Adds an Order” | 1.3.1 Create an Order table in the database | MUST | Yes |
|  |  | 1.3.2 Create an “Orders” tab in the GUI that contains a list of orders and an “Add Order” button | MUST | Yes |
|  |  | 1.3.3 Provide an order creation form with a save button connected to the database | MUST | Yes |
|  | 1.4 “A User Edits/Views an Order” | 1.4.1 Provide the option to click on a particular order number in the “Orders” tab and have it bring you to an order editable details page with a save button | MUST | Yes |
|  | 1.5 “A User Deletes an Order” | 1.5.1 Allow an order to have checkbox next to its entry in the “Orders” tab table to be checked and add a “Delete” button to the page | SHOULD | Yes |
|  |  | 1.5.2 Remove the specified deletion order from the database | SHOULD | Yes |
|  | 1.6 “A User Views the Summary of Outstanding Orders” | 1.6.1 Provide a table of all orders within the orders table in the “Orders” tab view. | MUST | Yes |
|  | 1.7 “A User Views Past Orders” | 1.7.1 Same as above; all orders should be present in the orders table, including past/finished orders. | MUST | Yes |
|  | 1.8 “A User Views a Summary of Customers” | 1.8.1 Create a “Customers” tab in the GUI that contains a list of customers. | MUST | Yes |
|  |  | 1.8.2 Create a “Customer” table in the database to allow the above table to be filled | MUST | Yes |
|  | 1.9 “A User Views a Summary of Salesmen” | 1.8.1 Create a “Salesman” tab in the GUI that contains a list of hired salesmen. | MUST | Yes |
|  |  | 1.8.2 Create a “Salesman” table in the database to allow the above table to be filled | MUST | Yes |
| 2  Velocity = 45 | 2.1 “A User Adds an Customer” | 2.1.1 Add an “Add” button to the Customer tab page | MUST | Yes |
|  |  | 2.1.2 Provide a Customer creation form with a save button connected to the database | MUST | Yes |
|  | 2.2 “A User Edits/Views a Specific Customer” | 2.2.1 Provide the option to click on a particular customer number in the “Customer” tab and have it bring you to an editable Customer details page with a save button | MUST | Yes |
|  | 2.3 “A User Deletes a Customer” | 2.3.1 Allow an Customer to have checkbox next to its entry in the “Customer” tab table to be checked and add a “Delete” button to the page | SHOULD | Yes |
|  |  | 2.3.2 Remove the specified Customer from the database | SHOULD | Yes |
|  | 2.4 “A User Adds an Salesman” | 2.4.1 Add an “Add” button to the Salesman tab page | MUST | Yes |
|  |  | 2.4.2 Provide a Salesman creation form with a save button connected to the database | MUST | Yes |
|  | 2.5 “A User Edits/Views a Specific Salesman” | 2.5.1 Provide the option to click on a particular salesman number in the “Salesman” tab and have it bring you to an editable Salesman details page with a save button | MUST | Yes |
|  | 2.6 “A User Terminates a Salesman” | 2.6.1 Allow an Salesman to have checkbox next to its entry in the “Salesman” tab table to be checked and add a “Delete” button to the page | SHOULD | Yes |
|  |  | 2.6.2 Remove the specified Salesman from the database | SHOULD | Yes |
|  | 2.7 “A User Manually Assigns an Order to a Salesman” | 2.7.1 In the order creation and view/edit forms, allow a salesman to be assigned to a particular order. This must assigned that salesman to that order. | MUST | Yes – This feature was removed in iteration 3 due to requirement changes |
|  | 2.8 “A User Changes Their Settings” | 2.8.1 Add setting tuples to the User table in the database | SHOULD | Yes |
|  |  | 2.8.2 Add a “Settings” tab to the GUI view with a form of changeable settings options | SHOULD | Yes |
|  |  | 2.8.3 Connect these setting changes to the database and have these settings reflected for that particular user. | SHOULD | No – This was partially implemented but moved to iteration 3 |
| 3  Velocity = 50 | 3.1 “A User Automatically Assigns an Order to a Salesman” | 3.1.1 In the order creation and view/edit forms, allow the “Automatically Assign Salesman” option to be chosen. | SHOULD | Yes |
|  |  | 3.1.2 Write an algorithm to find the least busy Salesman on the order’s due date, and assign him/her the order. | SHOULD | No – However, this feature was removed in iteration 3 due to requirement changes |
|  | 3.2 “A User Views Their Order Routes” | 3.2.1 Create a salesman Order page that lists only the orders assigned to the current user salesman. | MUST | Yes |
|  |  | 3.2.2 Provide a route button on that page which calculates the optimal route starting from the salesman’s house address through all of their orders | MUST | Yes |
|  |  | 3.2.3 Provide a Bing Maps representation of a particular Order’s route for each order in the route path. | SHOULD | No – This feature was removed before this functionality was added |
|  | 3.3 “A User Views Salesman Account Statistics” | 3.3.1 Provide a “Statistics” button in a particular Salesman’s view page that brings the user to a new page filled with that salesmen’s statistics. | SHOULD | Yes |
|  |  | 3.3.2 These statistics must be loaded from that salesman’s database table, and thus must be updated whenever a salesman performs an order. | SHOULD | Yes |

Note that this table does not contain the two new added user stories as they were not planned for this iteration, and were instead added after the iteration was under-way. Those user stories are detailed in depth in section 2, were both mandatory, and were both completed.

## 3.5 Source Control Server and Continuous Integration

The source control server we used for our project was Microsoft’s Team Foundation Server. This same tool also allowed us continuous integration functionality, and thus every member of our group was able to edit the website’s source code and have those changes immediately appear on the hosted website upon compilation.

## 3.6 Source Code Analysis

The application itself was written in C# using XAML as a GUI mark-up language. There are several qualities by which source code can be judged: coupling, cohesion, code readability, and code extensibility. Additionally, there may be usability problems and/or bugs related to the successful deployment of code. These qualities concerning our project code are described in greater detail here:

* Coupling:
* Cohesion:
* Code Readability:
* Code Extensibility:
* Usability Problems:
* Other issues:

## 3.7 Software Tests

For testing the application, we used two different methods. Firstly, for testing the GUI’s functionality, we scripted Sikuli test cases that provided a deep coverage of every user story. We also created a coded database-testing suite within the Team Foundation Server framework which comprehensively tests additions and removals of table information from every type of table in our database.

As such, there is technically 100% testing coverage for our website’s functionality using database reads/writes and user stories as coverage statistics. The GUI tests also act as a checker for the routing algorithm, ensuring that the correct routes are chosen through many separate test cases.

## 3.8 Future Release Plans

For future releases of the SalesRunner system, there are several additional features that our customers may be interested in that we did not have the time or resources to implement. These include the following:

* Giving veteran salesmen greater leeway for choosing particular orders to complete.
* Providing a manager the ability to manually assign an order to a salesman and sidestep routing completely.
* A greater variety of GUI-changing options in the settings tab.

Other possible features would be up to the customers to decide.

# 4.0 SalesRunner Updated Storyboard (Week 3)

The functionality of a software system can be described as an amalgam of user stories. These user stories are simply tasks or features which a user may do with the system. This section contains a detailed overview of all of the user stories which make up the SalesRunner 2012 system, and identifies which of these user stories will be released in the first release of the SalesRunner 2012 application. Each user story will be represented in a table “card”, and as a group they become the SalesRunner storyboard.

The user stories will be grouped into functional categories to ease browsing. User stories whose names are contained in a green box have already been implemented. User stories whose names are contained in a red box will not be implemented for this release. User stories highlighted in orange are user stories that have been removed or replaced. Finally, the finished stories in this table are labeled according to which iteration of development they were completed in. The weekly storyboard snapshots for weeks 1 and 2 of iteration 3 can be found in Appendix D.

## 4.1 Verification

|  |
| --- |
| A User Logs In – Iteration 1 |
| Precondition: None |
| **Steps:** The user must enter a valid username and password into the specified fields and press enter. |
| **Post Condition:** User is logged into the system. Depending on the user’s credentials (either a Manager or a Salesman user), there will be more functionalities offered to the user. Manager-type users have more system access than Salesman-type users. |
| **Priority**: High |

|  |
| --- |
| A User Logs Out – Iteration 1 |
| **Precondition:** A user is logged into the system |
| **Steps:** When inside the system, press the “Logout” button/tab on the top right corner of the screen. |
| **Post Condition:** The user loses access to the system, and is brought to the login screen. |
| **Priority**: Medium |

## 4.2 Order Management

|  |
| --- |
| A User Adds an Order – Iteration 1 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** When inside the system, press the “Orders” tab at the upper part of the screen. Then, press the “Add Order” button below the “Orders” tab. A form will appear, covering the other parts of the screen, leaving the tab menu at the top. The user can then fill in the specific fields on the form and the specifics of the purchases on the order. When the form is complete, the user would save the form by pressing the “Save” button at the top left part of the form. |
| **Post Condition:** Adds an Order table to the database. Also links the Order entity to a Customer and a Salesman entity. |
| **Priority**: High |

|  |
| --- |
| A User Edits/Views an Order – Iteration 1 |
| **Precondition: V**iews on the order could change depending on the user type. Managers can view/edit all orders while salesmen can only edit the orders they’re assigned to. |
| **Steps:** On the main orders page, clicking on an order’s “order number” will take you to the detailed order page for that order. When clicked, the filled-up order form for the specific entity will appear. For a manager user, the form is editable and the edit could be saved (“Save” button), while a salesman user could only view the form and not edit it. |
| **Post Condition:** When the manager decides to edit it and save the changes (“Save button”), the changes will be saved to the database. If a database error occurs, (ie. an integrity violation), changes are not saved and user is alerted. |
| **Priority**: High |

|  |
| --- |
| A User Deletes an Order – Iteration 1 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** On the orders screen, the user could select the forms that could be deleted by clicking the checkbox on the right corner of the entity. Multiple checkboxes could be selected. Then, the user could erase the selected orders by pressing the “Delete” button on the upper right side of the view orders. A confirmation prompt will appear to avoid accidental deletion. |
| **Post Condition:** Erases the selected Order entity, ensuring there is no database referential/key violation that will occur for a Customer or Salesman entity. |
| **Priority**: Low |

|  |
| --- |
| A User Views the Summary of Outstanding Orders – Iteration 1 |
| **Precondition:** Views could change depending on the user-type logged on to the system. A manager could see all of the outstanding orders while a salesman could only view the orders assigned to them. |
| **Steps:** View “Summary of Outstanding Orders” is the default view after logging in to the system. Alternatively, when the user is browsing another tab, clicking the “Orders” tab at the top left corner of the screen will bring the user to the interface. The order summary is now listed. |
| Post Condition: None |
| **Priority**: Low |

|  |
| --- |
| A User Views Past Orders – Iteration 2 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** When browsing the orders menu, the user will see a summary view of past completed orders recorded in the system (the view is similar to the outstanding orders summary view). |
| Post Condition: None |
| **Priority**: Low |

## 4.3 Customer Management

|  |
| --- |
| A User Adds a Customer – Iteration 2 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Click the “Customer” tab beside the “Orders” tab in the menu. It will show the user the summarized list of the customers recorded in the system. Clicking on the “Add Customer” button below the “Orders” tab will bring a form that could be filled out to add a customer to the system. The form that is brought up has several fields for the specific details of the customer. After filling in the form, a “Save” button could be clicked to register the customer to the system. |
| **Post Condition:** A Customer entity will be registered to the database. |
| **Priority**: High |

|  |
| --- |
| A User Edits/Views a Specific Customer – Iteration 2 |
| **Precondition:** A manager user could view and edit all the customer entities in the system, while salesmen could only view the customers assigned to them, either as a visit or a placed order. Either way, the user must be logged in. |
| **Steps:** Similar to the view/edit orders feature. When on the summary of the customer view, either the customer number or customer name is clicked to bring up the detailed view. A detailed view on the customer will be shown in the form similar to the “Add Customer” form. For managers, these forms are editable, while for salesmen the forms can’t be editable. |
| **Post Condition:** When the manager decides to edit it and save the changes (“Save button”), the changes will be saved to the database. Make sure no database integrity violation will occur. |
| **Priority**: High |

|  |
| --- |
| A User Deletes a Customer – Iteration 2 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** On the view summary of customers, click the checkbox in the right side of a customer entity. Multiple checkboxes could be selected by the user. Then after selecting some checkboxes, the “Delete” button could be pressed that deletes the selected entities to the system. |
| **Post Condition:** The deleted entities must be erased on the database. Make sure no database integrity violation will occur. |
| **Priority**: Low |

|  |
| --- |
| A User Views a Summary of Customers – Iteration 1 |
| **Precondition:** A manager user could view all customer entities in the system. A salesman could only view the customers assigned to them, either as a scheduled visit, or an order placed by that customer. |
| **Steps:** Click the “Customers” tab on the menu bar to bring this view. It will show the customers and its summary formatted as one straight line in the screen. |
| Post Condition: None |
| **Priority**: Low |

## 4.4 Salesmen Management

|  |
| --- |
| A User Adds a Salesman – Iteration 2 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** If not yet on the “View Summary of Salesmen”, click on the “Salesman” tab on the menu bar. Then, click the “Add Salesman” button at the top of the salesmen list. A form will appear for the user to fill up about the new salesman to be added. The new salesman number is assigned automatically. The form contains fields for the details of the salesman, which include: name, address, contact info, date added, status, position, experience level, current orders and notes. After filling in the form, press “Save” on the top left corner of the form to save the details of the newly added salesman. |
| **Post Condition:** The salesman entity must be saved on the database. |
| **Priority**: High |

|  |
| --- |
| A User Edits/Views a Specific Salesman – Iteration 2 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** On the View Salesmen Summary, select the salesman that needed to be viewed or edited. Then, a form will appear showing all the description about the salesman. The forms are editable, and clicking “Save” button saves the changes made. |
| **Post Condition:** Save the changes made to a specific salesman when the “Save” button is pressed. |
| **Priority**: Low |

|  |
| --- |
| A User Views a Summary of Salesmen – Iteration 1 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Clicking the “Salesman” tab on the menu bar shows the summary of salesmen in the system. The salesmen are summarized and formatted in separate lines. |
| Post Condition: None. |
| **Priority**: Low |

|  |
| --- |
| A User Terminates a Salesmen – Iteration 2 |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Clicking the “Salesman” tab on the menu bar shows the summary of salesmen in the system. The user could select a salesman to terminate, and save their changes. |
| **Post Condition:** A Salesman entity has its status changed to “terminated”. |
| **Priority**: Medium |

## 4.5 Salesman Routing

|  |
| --- |
| A User Manually Assigns an Order to a Salesman – Iteration 2 – REMOVED |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Click on the “Orders” tab in the menu bar to show the summary of outstanding orders to be completed. Then edit and press the “Save” button to save the changes. |
| **Post Condition:** Saved changes will be registered in the database. Also, the Order and Salesman entity should be linked via a foreign key. Any changes to the foreign key (if the salesman is changed) must not violate any integrity of the database. |
| **Priority**: Medium |

|  |
| --- |
| A User Automatically Assigns an Order to a Salesman – Iteration 3– REMOVED |
| **Precondition:** The user must be logged in as a Manager. |
| **Steps:** Click on the “Orders” tab in the menu bar to show the summary of outstanding orders to be completed. Then edit the form (see Edit Form feature). Then on the “Assigned to” status, change the setting to be automatic. Then press the “Save” button to save the changes. |
| **Post Condition:** Saved changes must be registered in the database. The application layer for priority and routing distance calculation must assign the order to a specific salesman. Order entity and Salesman entity must be linked via foreign key. There should be no integrity violation in the database. |
| **Priority**: High |

|  |
| --- |
| A User Views Their Order Routes – Iteration 3 – REMOVED |
| **Precondition:** Salesman user must be logged on to the system. |
| **Steps:** Click on the Orders tab to see the summary of outstanding orders that the salesman needs to fulfill. Then press the “Get Route” button on the top right corner of the summary form. The routing application layer will calculate the route for the next location order that needs to be done. |
| **Post Condition**: None |
| **Priority**: High |

## 4.6 Extra Features

|  |
| --- |
| A User Views Their Account Statistics – Iteration 3 |
| **Precondition:** A user must be logged on to the system. |
| **Steps:** As a manager: the “Salesman” page is clicked and a salesman is chosen by clicking on their name or number. The detailed page for that salesman is brought up and the “stats” button at the bottom is clicked. A form containing sales history, customer history, amount of forms sold, the number of customers sold to, and net sales is brought up. These forms can be sorted by month, or all time.  As a salesman: the “My Stats” tab is clicked, which brings up the same form as when “stats” is clicked as manager, but the stats for the currently logged in salesman can only be shown. |
| **Post Condition**: None |
| **Priority**: Low |

|  |
| --- |
| A User Changes Their Settings – Iteration 2 |
| **Precondition:** A user must be logged on to the system. |
| **Steps:** Click on the settings tab and manipulate any user settings deemed desirable. Important settings include changing the user’s password, changing address/contact/email fields, or perhaps even changing the colour scheme of the system for when they log in. The specific settings will be discussed in a later iteration. |
| **Post Condition:** The user’s settings will remain with them every time they access the system; the user’s database attributes will be changed to accomplish this. |
| **Priority**: Low |

## 4.7 New Features

|  |
| --- |
| A User Assigns Themselves (Claims) an Order – Iteration 3 |
| **Precondition:** A salesman must be logged on to the system. |
| **Steps:** User navigates to the Routing tab. A pop-up appears, asking whether the source route should be the salesman’s home or from the last order they completed. If the user presses home, the routing algorithm uses the user’s home address, and if the user pressed Last Order Visit, the routing algorithm uses the last completed order’s visit address for the “From” field in routing.  The routing algorithm finds the most optimal order for that salesman to undertake (via the calculation lowest return value for (distance to destination) – (order priority level)), and gives the user the option to claim or cancel the order. If the salesman claims the order, they have assigned themselves to that order. Otherwise, the user is returned to the order page. |
| **Post Condition:** A salesman claims their chosen order and can now fulfill that order. |
| **Priority**: High |

|  |
| --- |
| A User Views/Edits Their Claimed Order Details – Iteration 3 |
| **Precondition:** A salesman must be logged on to the system and has claimed an order. |
| **Steps:** User navigates to the order tab. The user’s previously claimed order details will appear, showing a Bing Maps route from their source location to the order’s visit location, as well as several fields which can be viewed/edited. These include adding items that were successfully sold to the customer (upon order completion), and changing the order’s State (Complete, Incomplete, or Delayed). The user solidifies the order when they set the state to complete; this updates the order completion date and all order settings become finalized. |
| **Post Condition:** A salesman claims their chosen order and can now fulfill that order. |
| **Priority**: High |

# **5.0 Detailed Member Efforts**

**William Lancaster** – Wrote the iteration 3 report and iteration 2 team retrospective, aided in the implementation of the routing page, and headed the Sikuli GUI test creation process.

**Wesley Henderson** – and aided in the creation of Sikuli GUI test cases.

**Eliezer Mar Manarang** – and aided in the creation of Sikuli GUI test cases.

**Egor Gerasimenko** – and headed in the creation of database test cases.

**Mike Badmington –** and aided in the creation of Sikuli GUI test cases.

# 6.0 References

[1] Google Inc. “Google Maps”. Retrieved on February 11, 2012 from the Internet: <http://maps.google.com/>

[2] MySQL. “MySQL :: The world’s most popular open source database”. Retrieved on February 11, 2012 from the Internet: <http://www.mysql.com/>

[3] Microsoft Corporation. “Visual C#”. Retrieved on February 26, 2012 from the Internet: <http://msdn.microsoft.com/en-us/vstudio/hh388566>

[4] Microsoft Corporation. “XAML”. Retrieved on March 7, 2012 from the Internet: <http://msdn.microsoft.com/en-us/library/ms752059.aspx>

[5] Microsoft Bing. “Bing Maps”. Retrieved on March 7, 2012 from the Internet: <http://www.bing.com/maps/>

# Appendix A – Customer Critique

This section contains the critique received by our customer group for both this report and our working application. Any questions that are raised are answered below them in *italic* font.

Critique of Iteration 3 Report and Application

Mark Mullen

**Report Critique**

* Requirements changed was implemented as discussed
* User stories affected by requirements change are accurate
* Work Completed correct
* Release task completion overview table is good
* At time of critique, Source Code Analysis was not yet complete
* Storyboard good

**Application Critique**

* At time of usage, I was unable to select any items in the database when making an order
* When entering incorrect information for a salesman, there is error checking to prevent a false entry (good) but a popup box informing me of my mistake would be nice
* Delete does not work on customer window
* Different views are nice
* Logout functionality is good
* Having placeholders in settings tab for future features is good

# **Appendix B –** Iteration 2 Retrospective

**Facilitator**: William Lancaster  
**Minutes**: 50 minutes  
  
**What Went Well?**

The report was completed early and was of high quality, earning a perhaps “too short” critique from our customers. At the very end of the iteration, a very good amount of work had been completed on the application, complete with unit tests and continuous integration.  
  
**What Could Have Been Improved?**

#### Communication

Once again, communication proved difficult outside of class-assigned meetings. Getting responses to e-mails is hard and often communication is seemingly closed off completely without finding members in person.

#### Division of Work

Some members were assigned work that did not get completed for the iteration, and others were forced to take on extra responsibility to ensure a working prototype by the end of the iteration. This implies an unfair division of work not from the outset, but in some member’s unwillingness to complete their work forcing others to work more than was planned.

#### Consistency of Progress

Once again much of the progress on the iteration came in quickly at the end rather than consistently throughout. This, however, can be attributed first and foremost to the heavy workload we have in other courses, but still presents a problem when it is apparent near the due date that certain members cannot finish their portion of the workload and others must bear that load, as mentioned in the previous section.

**Affinity Mapping: What category will be improved?**  
  
The group has voted that the category in most need of repair is the inconsistency of work progress. In the previous iteration, the team once again ended up finishing an inordinate amount of the work right before the scheduled due date. This also caused some members who had finished their work to work on other’s assigned tasks simply to ensure the work was completed.

#### Root Cause of Issues and Plan of Action

A lack of initial iteration planning coupled with a lack of task completion enforcement was the main cause of this problem. Compounding these issues is the large amount of course work all team members are tasked with for others courses; finding time to work on a project whose deadline is relatively far in the future is rather difficult.

Our plan to remedy this issue is to create a more solid weekly plan for this iteration, and to check the progress every team member has made towards their assigned tasks during every Monday meeting through the iteration. Without a proper disciplinary incentive aside from suboptimal peer review scores, a consistent nagging will have to suffice as the major incentive to finish work on time

# **Appendix C – Iteration 3 Meeting Details**

This section details the meeting notes written during our group meetings from most recent to least recent. These notes are in a rather rough format, as they were written while in mid-discussion, and regardless were able to properly serve their purpose.

**Wednesday, April 4, 2012 (50 minutes)**

Discussed with the group about iteration progress, especially the changes in the GUI aspects of the system:

* Difference in Salesman and Manager views of orders
* Additional Buttons (Claim and Confirm) for Salesman
* Changeable aspects of order entities in different views
* Some bugs encountered are also discussed.

**Monday, April 2, 2012 (15 minutes)**

Discussed with the group about iteration progress

**Wednesday, March 28, 2012 (90 minutes)**

Discussed with the group again on how to implement routing and order assigning.  
  
***New Assign salesman AND Routing Use Case***:

**Rough Description**:

A user assigns an order to the salesman closest. Needs confirmation to claim the order. Provides routing using External Maps API. Could change the order state after completion:

**Precondition**:

A salesman must be logged in. No orders must be assigned yet.

**Postcondition**:

Salesman finishes an order and changes its state. Then this use case loops again.

**Main Path**:

a) Salesman presses the Order tab (or default login).

b) A pop-up appears. It ask whether the source route is home or last order visited

c1) If User presses Home, the routing algorithm uses the user’s home address

c2) If User presses Last Order Visit, the routing algorithms uses the last order visit address.

d) The routing algorithm ranks the nearest order based on distance (if there is one) [with priority] for the current day, and suggests it to the user as the current order to be claimed. State about the order will be shown (immutable) with the map together with the vehicle routing direction. View is similar to the View/Edit Order for Salesman Storyboard, but moved to View Order Summary for Salesman Storyboard.

e) The user claims the order via pressing a Claim button. Order state about salesman will be changed to the user salesman id/username. The order can’t be used for routing for other salesman. Once claimed, the order state changes from Unassigned to In Progress.

f) The order state could be partially changed. Changes include:

* Adding items that are sold
* Changing its state from In Progress to:
* Completed (will remove the order from the current order table access, will mark the item to be completed) Only Manager could access completed Orders, but most fields are immutable.
* Incompleted (will remove the order from the salesman table, will mark the item to be completed) Only Manager could access completed Orders, but most fields are immutable.
* Delay (will move the order to next day’s current orders, also change priority) - Will make the state Unassigned again.

g) The user selects ‘Complete’ button to permanently save the order’s state. Completion Date will also be updated.

h) Refreshes the order page, start at (b).

**Alternate Path**:

a) After (d) from main path. If user doesn’t claim it, the page will refresh on every Order button press similar to (b).

b) If the order got claimed by another salesman (see (e) on main path). The Order page will be refreshed, showing another order due to routing algorithm.

Notes about the changes: - THIS COULD BE USED FOR ADD/CHANGE REQUIREMENT

* View Summary Orders - Salesman View - will be changed drastically, include detailed order page, and map. Will also include claim button and complete button (replaces the claim button). Also a refresh button - has the event that also being handled by the Orders tab event handling.
* Edit/View Order - Salesman View - REMOVED - has been moved to View Summary Orders Salesman storyboard. Some of the variables will now be immutable depending on the Order if it is claimed or not yet claimed.
* Auto Assign Salesman and Routing- uses a 2D matrix (salesman - order dimension) for routing. Each refresh only handles the salesman row of order distances-priority for the specific user. Both feature are now combined.
* A ‘Claim’ event must be added
* Manual Assign Salesman – REMOVED
* Add Order Manager
* fields that are removed:
  + Distance from Salesman
  + Difficulty
  + Reorder Chance
* Other fields that are changed
  + Assigned to: Immutable - Unassigned if no one claimed it yet (if state is Unassigned).
  + Name of the assigned salesman (if state is In Progress, Completed, Incomplete)
* New Possible Order States:
  + Unassigned
  + Completed
  + Incomplete
  + Delay (will be changed to Unassigned quickly or not be shown at all)
* View/Edit Order Manager
  + If State is unassigned, all states could be still changed
  + If State is either Complete or Incomplete:
  + Only Items table could be changed. Everything is immutable (Salesman entity)
  + Remove - CurrentStatus and Experience(Level)

**Monday, March 26, 2012 (90 minutes)**

Meeting with the whole group discussing the future plans for the iteration

What should we need to do before the iteration finishes:  
    a) Finish the routing (using maps API, salesman assign algorithm, routing algorithm, other parts that will be affected)  
    b) Finish the styles  
    c) Finish the delete options for Salesman and Customer  
    d) Retrospective meeting [report]  
    e) Decide what should we add or change for requirements [part report]  
    f) System analysis [report]  
    g) Release schedule for the past project release [report]  
    h) Continuous intergration  
    i) Salesman View  
    j) Detailed member effort [report]  
    k) Weekly meeting snapshot [report]

**Assigning parts**:  
a) Using maps API - Egor  
b) Routing algorithm: (distance - priority, the lower the better) - Wesley  
c) Autoassign algorithm: (number of orders on the day, even it out) - Elmar  
d) Salesman Statistics (visited\_successful, visited\_failed, cancelled, in progress) - based on order state - Elmar  
e) Adding items to database and orders - Postponed  
f) Settings/Styles – Mike

**Routing Use Cases - to confirm what we have planned**

**\*\*\*Use Case Name: Assigning an Order to a Salesman\*\*\***  
**Description**: A Manager could assign an order to a salesman  
**Precondition**: A salesman entity and a customer entity should exist in the system  
**Postcondition**: The order assigned will be seen by the salesman, and could be routed by the salesman  
**Main** **Path**:

a1) The Manager creates an order and assign an order to a customer  
a2) The Manager selects a preexisting order in the system  
b) The Manager assigns a salesman either manually or autoassign  
c) If auto assign, let the system handle assigning via the salesman’s busyness and skill level statistics

Alternate path:  
None

**\*\*\*Use Case Name: Salesman Routing\*\*\***  
**Description**: A salesman could route from a source to a customer destination  
**Precondition**: An order exists in the system, and assigned to a salesman  
**Postcondition**: System routes the salesman using a routing algorithm (also with the help of an external routing map API[1]).  
**Main Path**:

a) The salesman views the routing page of the system and views the orders to be visited for the day (past unsuccessful orders and the day’s orders)  
b) The salesman presses the route button.  
c) The system will ask whether the source location is the last finished order location (stored in the system) or home. (If there’s a geolocation, this is not needed)  
d) The system uses a routing algorithm implemented by finding the distance of each destination from the source and apply priority factor of each order to find the ranking in where to go. The system picks the first in the ranking and shows the routing to the salesman via the map API.  
e) When order is finished, the order will be removed from the routing table.  
Alternate path:  
a) The salesman skips in going to the first ranked order via pressing a skip button.  
b) The system picks the second in the ranking

**Wednesday, March 21, 2012 (30 minutes)**

* Had an informal meeting.
* Discussed about planning to make Weekly deadlines for each group member for the iteration.
* To make a checklist of what should we do for the iteration 3 before Friday

# Appendix D – Storyboard Progress

**Storyboard for week 1**

#### Verification

|  |
| --- |
| A User Logs In |
| A User Logs Out |

#### Order Management

|  |
| --- |
| A User Adds an Order |
| A User Edits/Views an Order |
| A User Deletes an Order |
| A User Views the Summary of Outstanding Orders |
| A User Views Past Orders |

#### Customer Management

|  |
| --- |
| A User Adds a Customer |
| A User Edits/Views a Specific Customer |
| A User Deletes a Customer |
| A User Views a Summary of Customers |

#### Salesmen Management

|  |
| --- |
| A User Adds a Salesman |
| A User Edits/Views a Specific Salesman |
| A User Views a Summary of Salesmen |
| A User Terminates a Salesmen |

#### Salesman Routing

|  |
| --- |
| A User Manually Assigns an Order to a Salesman |
| A User Automatically Assigns an Order to a Salesman |
| A User Views Their Order Routes |

#### Extra Features

|  |
| --- |
| A User Views Their Account Statistics |
| A User Changes Their Settings |

**Storyboard for week 2**

#### Verification

|  |
| --- |
| A User Logs In |
| A User Logs Out |

#### Order Management

|  |
| --- |
| A User Adds an Order |
| A User Edits/Views an Order |
| A User Deletes an Order |
| A User Views the Summary of Outstanding Orders |
| A User Views Past Orders |

#### Customer Management

|  |
| --- |
| A User Adds a Customer |
| A User Edits/Views a Specific Customer |
| A User Deletes a Customer |
| A User Views a Summary of Customers |

#### Salesmen Management

|  |
| --- |
| A User Adds a Salesman |
| A User Edits/Views a Specific Salesman |
| A User Views a Summary of Salesmen |
| A User Terminates a Salesmen |

#### Salesman Routing

|  |
| --- |
| A User Manually Assigns an Order to a Salesman |
| A User Automatically Assigns an Order to a Salesman |
| A User Views Their Order Routes |

#### Extra Features

|  |
| --- |
| A User Views Their Account Statistics |
| A User Changes Their Settings |

Note: The two new user stories mentioned in section 2 were created and tackled within the final week of implementation, and are thus not present in the two first weeks of iteration 3.