CSc 4350: Software Engineering

Fall 2016

Team Phoenix

Resort Management System

Fergus Kelley

Andrew Truong

Ryan Ocampo

Deividas Rutkauskas

Viraj Shah

Document #5 — Project Rationale

October 13th, 2016

Requirements Traceability Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entry # | Para # | PRMS Requirements Traceability Matrix | Type | Use Case |
| 1 | 2.0 | The PRMS shall provide a graphic user interface (GUI) with tabs. | SW |  |
| 2 | 2.0 | Each tab of the interface shall profile functionality for employees to interact with the PRMS. | SW |  |
| 3 | 2.0 | When the employee clicks on a tab, the PRMS shall display a new interface with the appropriate functionality for that tab. | SW |  |
| 4 | 3.0 | The PRMS shall provide a tab of the interface for creating and managing employee profiles | SW |  |
| 5 | 3.0 | The employee profiles shall be stored in a local database. | SW |  |
| 6 | 3.0 | Each profile shall contain information about the employee. | SW |  |
| 7 | 3.0 | Each profile will contain information about the employee’s access to the system. | NTH |  |
| 8 | 3.1 | The PRMS shall provide an interface where the new employee profiles are created. | SW |  |
| 9 | 3.1 | Each employee profile shall contain the employee’s first and last name, job title, username, and password. | SW |  |
| 10 | 3.1 | The PRMS will provide functionality for suitably authorized employees to create new employee profiles by inputting all the necessary information | SW | UC2\_EmployeeProfile |
| 11 | 3.1 | The PRMS will provide functionality for suitably authorized employees to modify the information contained in already existing profiles. | SW | UC2\_EmployeeProfile |
| 12 | 3.1 | The PRMS will provide functionality for suitably authorized employees to remove existing employee profiles | SW | UC2\_EmployeeProfile |
| 13 | 3.2 | The PRMS will limit access by way of a username and password | SW | UC1\_UserLogin |
| 14 | 3.2 | The employee profile will contain information about which interface tabs are available to each employee based on the employee’s title. | SW |  |
| 16 | 3.2 | The PRMS will log employee interactions with the PRMS. | NTH |  |
| 17 | 4.0 | The PRMS shall allow the management and reservation of the resort’s hotel rooms. | SW | UC3\_ManageHotelRooms |
| 18 | 4.1 | The PRMS shall provide a tab of the user interface for managing the resort’s hotel rooms. | SW | UC3\_ManageHotelRooms |
| 19 | 4.1 | The PRMS shall provide functionality for describing the number of floors of the hotel, and the number of rooms on each floor. | SW | UC3\_ManageHotelRooms |
| 20 | 4.1 | The PRMS shall assign each room a number based on which floor of the hotel the room is located. | SW |  |
| 21 | 4.1 | The PRMS shall provide functionality to enter and modify default attributes about each room. | SW | UC4\_ModifyHotelRoom |
| 22 | 4.1 | Each room profile shall have attributes and a list of the rooms inventory. | SW | UC4\_ModifyHotelRoom |
| 23 | 4.1 | For each item in the room, the room inventory shall keep track of the name, quantity, expected quantity, and whether the item is consumable. | SW |  |
| 24 | 4.1 | The PRMS shall keep track of the number of each item in the rooms inventory that has been replaces. | SW |  |
| 25 | 4.1 | The room profiles shall be stored in a local database. | SWC |  |
| 26 | 4.2 | The PRMS shall provide a tab of the user interface that provides functionality for employees to create and modify reservations for each room. | SW | UC5\_CreateReservation |
| 27 | 4.2 | The room reservations shall be stored in a local database. | SWC |  |
| 28 | 4.2 | Each reservation shall contain attributes. | SW |  |
| 29 | 4.2 | The reservation’s billing information shall contain fields. | SW |  |
| 30 | 4.2 | The PRMS shall allow employees to query the list of rooms by the room’s attributes and whether a reservation exists for a given timespan. | SW | UC5\_CreateReservation |
| 31 | 4.2 | The PRMS shall allow employees to create reservations for any room that does not have a reservation for a given time span. | SW | UC5\_CreateReservation |
| 32 | 4.2 | The PRMS shall require the employee to enter valid information for all the fields in the room reservation information. | SW |  |
| 33 | 4.2 | The PRMS shall allow employees to modify a reservation’s billing information by adding or removing additional charges. | SW | UC6\_Billing |
| 34 | 4.2 | The PRMS shall allow employees to modify a reservations timespan to extend or shorten a guest’s stay. | SW | UC5\_CreateReservation |
| 35 | 4.2 | The PRMS shall allow employees to output an itemized invoice containing all information about the room, including the calculated cost of the reservation based on the price per night and timespan. | SW | UC6\_Billing |
| 36 | 4.3 | The PRMS shall provide a tab of the user interface that provides functionality for employees to manage room maintenance and inventory. | SW | UC7\_Maintenance |
| 37 | 4.3 | The PRMS shall allow the employee to note modify the number of items in the room inventory to indicate whether replacements are needed. | SW | UC7\_Maintenance |
| 38 | 4.3 | The PRMS shall provide an interface to note when the room has been cleaned or the inventory has been replaced and update the necessary fields in the room profile. | SW | UC7\_Maintenance |
| 39 | 4.3 | The PRMS shall allow the employee to output a report of all rooms which haven’t been cleaned in a given time period, rooms which have maintenance notes, or rooms with insufficient inventory. | SW | UC7\_Maintenance |
| 40 | 5.0 | The PRMS shall allow the management and reservation of the resort’s events rooms. | SW |  |
| 41 | 5.1 | The PRMS shall provide a tab of the user interface for managing the resort’s events rooms. | SW | UC8\_ManageEventRooms |
| 42 | 5.1 | The PRMS shall provide functionality for describing the number of events rooms in the resort. | SW | UC8\_ManageEventRooms |
| 43 | 5.1 | The PRMS shall provide functionality to enter and modify default attributes about each events room. | SW |  |
| 44 | 5.1 | Each events room shall have attributes. | SW |  |
| 45 | 5.1 | The events room profiles shall be stored in a local database. | SWC |  |
| 46 | 5.2 | The PRMS shall provide a tab of the user interface that provides functionality for employees to create and modify bookings for events rooms. | SW | UC9\_CreateBooking, UC6\_Billing |
| 47 | 5.2 | Each events room reservation shall contain attributes. | SW |  |
| 48 | 5.2 | The booking billing information shall contain fields. | SW |  |
| 49 | 5.2 | The PRMS shall allow employees to query the list of events rooms by the room’s attributes and whether a booking exists for a given timespan. | SW | UC9\_CreateBooking |
| 50 | 6.0 | The PRMS shall provide functionality for employees to manage the various services that are necessary to provide to the guests. | SW |  |
| 51 | 6.1 | The PRMS shall provide a tab of the user interface that provides functionality for employees to create and modify service orders. | SW | UC11\_CreateOrder, UC12\_ModifyOrder |
| 52 | 6.1 | Each order shall contain the attributes. | SW |  |
| 53 | 6.1 | The PRMS shall allow the employee to create a new order. | SW | UC11\_CreateOrder |
| 54 | 6.1 | For a new order, the state of the order shall default to “ordered” and the date and time expected shall default to the current time. | SW |  |
| 55 | 6.1 | The PRMS shall ensure that the employee has selected or created an invoice for the order provided. | SW |  |
| 56 | 6.1 | The PRMS shall display an interface for entering in the various data for the type of order the employee has selected | SW | UC11\_CreateOrder |
| 57 | 6.1 | The PRMS shall allow employees to view a list of orders sorted by time expected and filterable by current state | SW | UC13\_OutputOrders |
| 58 | 6.1 | The PRMS shall allow the employee to update the state of the order. | SW | UC12\_ModifyOrder |
| 59 | 6.1 | Once the state of the order is changed to “delivered,” the invoice associated with the order shall be updated with the name and cost of the order. | SW |  |
| 60 | 6.1 | Orders shall be stored in a local database. | SWC |  |
| 61 | 6.2 | The PRMS shall provide an interface that provides functionality for managing the hotel restaurant. | SW |  |
| 62 | 6.2 | The PRMS shall allow employees to create and modify a list of restaurant items available for room service and table service. | SW | UC14\_ManageRestaurant |
| 63 | 6.2 | Each restaurant item shall contain attributes. | SW |  |
| 64 | 6.2 | The PRMS shall allow employees to create and modify a list of tables available for seating at the restaurant | SW | UC14\_ManageRestaurant |
| 65 | 6.2 | Each table shall contain attributes. | SW |  |
| 66 | 6.2 | When creating an order for room service or table service, the PRMS shall require that the employee select one or more items and select or create an invoice for billing. | SW | UC11\_CreateOrder |
| 67 | 6.2 | If creating an order for table service, the PRMS shall require that the employee select a table number from the list of restaurant tables. | SW | UC11\_CreateOrder |
| 68 | 6.3 | The PRMS shall allow employees to create and modify a list of options available for catered meals. | SW | UC14\_ManageRestaurant |
| 69 | 6.3 | Each catered option shall contain attributes. | SW |  |
| 70 | 6.3 | When creating an order for catered meals, the PRMS shall require that the employee select a conference room booking for delivery and billing, as well as enter the number of servings required. | SW | UC11\_CreateOrder |
| 71 | 6.4 | The PRMS shall provide an interface that provides functionality for creating general orders. | SW | UC11\_CreateOrder |
| 72 | 6.4 | When creating a general service order, the PRMS shall provide a text box for entering specific information about the order. | SW | UC11\_CreateOrder |
| 73 | 7.0 | The PRMS shall provide a tab of the user interface that provides functionality for employees to easily query the various databases used by the software. | SW | UC15\_Reports |
| 74 | 7.0 | The PRMS shall provide interfaces for returning information in any given timespan. | SW | UC15\_Reports |

Test Cases

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | User Login |
| Tester | Ryan Ocampo |
| Input | I entered the combination of an incorrect username and password, and then I entered a combination of a correct username and password. |
| Oracle | For a correct combination user name or password, text that says “Login Successful!” is displayed and the user interface access tabs are enabled for user to select. For an incorrect combination of user name or password, text that says “Incorrect user name and password; try again” is displayed and tabs remain disabled. |
| Log | Results match expected output. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | User Logout |
| Tester | Viraj Shah |
| Input | For checking the logout functionality, a user must first be successfully authenticated into the application. If the user credentials are correct, when a user navigates to the Login Screen, the only UI Element that is enabled is the logout button. By pressing the logout button, the event handler logout is triggered which clears the Login and Password Text Fields and changes the submit button title text from Logout to Login. |
| Oracle | For a successful User Logout, the submit button title text is changed from “Logout” to “Login”, the Username and Password Text Fields are cleared, and the additional tabs are made unavailable for selection.  For an incorrect User Logout scenario, pressing the Logout Button will not change any of the UI elements and the Application tabs will still be available for navigation. |
| Log | Results match expected output. |

Continued on next page.

Continued from previous page.

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Create Employee |
| Tester | Deividas Rutkauskas |
| Input | By default, the "Create" radio button is selected, but in case it is not, the user selects it.  The user enters information for each field required to create an employee, namely "first name", "last name", "job title", "username", "password" and "confirm password."  The user then clicks the "Apply button." |
| Oracle | The employee table is correctly updated to include the new Employee with the information provided by the user.  If the user doesn't provide the appropriate information in the relevant fields, a message appears saying that a field or fields have been left blank.  If the password confirmation doesn't match the password field, a message appears that the provided passwords don't match. |
| Log | Results match expected output. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Modify Employee |
| Tester | Fergus Kelley |
| Input | An employee is selected in the table containing a list of employees.  The "Modify" radio button is selected.  To modify the employee's first or last names, the values in each field is changed.  To modify the employee's job title, a new value is selected from the drop down box.  To modify the employee's password, a new value is entered in the password field and the same value is entered into the confirm password field.  To apply the changes made in the field, the apply button is pressed. |
| Oracle | If the password and confirm password fields do not match, "password fields do not match" is displayed and modifications are not applied.  The employee table and database are correctly updated to reflect the applied modifications to the selected employee. |
| Log | Results match expected output. |

Continued on next page.

Continued from previous page.

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Remove Employee |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will enter the text fields with the name of the specific employee by using characters. After inputting data, the user will click on the remove button. |
| Oracle | It is expected that the specific employee that is requested from the user to be removed from the database. |
| Log | The output was a success. The message of successfully removing the employee appeared and the employee is removed from the database. |

Test Case Rationale

In order to sufficiently test the current build of our software, we decided to use a vertical integration testing strategy. We selected five vertical slices of our software’s current functionality: logging in, logging out, creating an employee profile, modifying an existing employee profile, and removing an existing employee profile. For each component, we developed a simple test case that corresponds to a small section of source code, allowing errors to be easily found and corrected. Additionally, by using vertical integration testing, we were able to quickly develop just a portion of our overall system and ensure that it was working correctly before creating full horizontal layers. A possible trade off of vertical integration testing is the risk of incorrect system design decisions, however, we have not yet encountered any such problems.

In performing testing, we were pleased to find that in each test case our software passed.

Constructive Cost Model (COCOMO)

Using the Intermediate COCOMO model with Organic Program Complexity:

***EAF Cost Drivers***

RELY: 1

DATA: 0.94

CPLX: 0.7

TIME: 1

STOR: 1

VIRT: 0.87

TURN: 0.87

ACAP: 1

AEXP: 1

PCAP: 1

VEXP: 1

LEXP: 1

MODP: 0.89

TOOL: 1.24

SCED: 1

**Calculated EAF = 0.55**

***COCOMO Calculation***

So according to COCOMO, the software would take **13.6** person-months of effort over **6.7** months of development time.

***COCOMO Rationale***

Using COCOMO, we are able to roughly predict how much time and effort would be needed to create our software. This allows us to estimate the amount needed to charge clients. For example, at $6000 per person-month, this software would cost us $81600 to develop.

Work Schedule Diagram

Gantt Chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Start Date:  August 22, 2016** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Week:** | **1** |  |  |  |  | **2** |  |  |  |  | **3** |  |  |  |  | **4** |  |  |  |  | **5** |  |  |  |  |
|  | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** |
| **Document 1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Document 2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Document 3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus: Gantt chart, FP, RTM |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Continued from previous page.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Week:** | **6** |  |  |  |  | **7** |  |  |  |  | **8** |  |  |  |  | **9** |  |  |  |  | **10** |  |  |  |  |
|  | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** | **M** | **T** | **W** | **T** | **F** |
| **Document 3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus: Gantt chart, FP, RTM |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj: Use cases and diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Document 4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus: Gantt chart, WSD, RTM |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew: Object design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan: Object design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas: Object design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj: Object design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Document 5** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus: Rationale, Gantt, WSD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Document 6** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus: Source Code, Test Case, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RTM, Gantt, WSD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew: Source Code, Test Case, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| COCOMO |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan: Source Code, Test Case |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas: Source Code, Test Case |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj: Source Code, Test Case |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Dictionary

*Embedded database*: a database that is used within a single application. This is opposed to a client-server database which involve two different software applications communicating.

*Graphical User Interface* (abbreviated *GUI*): A program that allows the user to interact with the computer using icons and other visual indicators.

*Local Database*: a collection of digital indexed information that can be searched, referenced, changed, compared or otherwise manipulated. A local database is stored on the same computer that is used to access it.

Change Log

**Date and Time:** October 23, 2016 — 9:45 PM

**Team member:** Fergus Kelley

**Description:** Initial creation of document

**Version:** 1 — Current version accumulating changes

**Date and Time:** October 24, 2016 — 10:51 PM

**Team member:** Fergus Kelley

**Description:** Updated RTM, WSD, Gantt, and title page. Added test cases by Ryan, Andrew, Viraj, and Deividas. Added COCOMO by Andrew. Added rationale for test cases and COCOMO.

**Version:** 1 — Current version accumulating changes