CSc 4350: Software Engineering

Fall 2016

Team Phoenix

Resort Management System

Fergus Kelley

Andrew Truong

Ryan Ocampo

Deividas Rutkauskas

Viraj Shah

Final Report

December 1st, 2016

Table of Contents

[Introduction 3](#_Toc468315369)

[Requirements Traceability Matrix 4](#_Toc468315370)

[Use Cases and Sequence Diagrams 9](#_Toc468315371)

[Category Interaction Diagram 33](#_Toc468315372)

[Object Design 34](#_Toc468315373)

[Test Cases 40](#_Toc468315374)

[Project Rationale 46](#_Toc468315375)

[Function Point Cost Analysis 47](#_Toc468315376)

[Constructive Cost Model (COCOMO) 48](#_Toc468315377)

[Prototype 49](#_Toc468315378)

[Project Legacy 54](#_Toc468315379)

[Work Schedule Diagram 55](#_Toc468315380)

[Gantt Chart 56](#_Toc468315381)

[Dictionary 59](#_Toc468315382)

[Resumes 60](#_Toc468315383)

[Fergus Kelley 61](#_Toc468315384)

[**Andrew An Dang Truong** 62](#_Toc468315385)

[Ryan Ocampo 63](#_Toc468315386)

[Deividas Rutkauskas 64](#_Toc468315387)

[**Viraj Shah** 66](#_Toc468315388)

[User Guide 68](#_Toc468315389)

[Installation and setup 68](#_Toc468315390)

[Employees 68](#_Toc468315391)

[Event Rooms 69](#_Toc468315392)

[Hotel Maintenance 69](#_Toc468315393)

[Hotel Reservations 69](#_Toc468315394)

[Event Bookings 69](#_Toc468315395)

[Invoices 69](#_Toc468315396)

[Orders 69](#_Toc468315397)

[Restaurant 69](#_Toc468315398)

# Introduction

When we formed Team Phoenix for this semester project, we want to pick a topic that would challenge our critical thinking and problem solving skills, while at the same time resembling a project we might encounter in our careers. We decided to create a piece of resort management software because it would require careful thinking to fully understand the needs of hotel guests and employees. From there, we decided to include additional tools for employee to manage other aspects that might be necessary at a resort, such as conference rooms and a restaurant. We want to explore the complexity of the business operations and create a piece of software that is both powerful and easy to use.

We are targeting the software to be used by the hotel employees to manage the operations of the resort. The software can be utilized on any contemporary operating system, such as Windows, Mac OS, or Linux since it will be developed using the Java language. A local database will be used for all persistent data necessary for the software.

# 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 | UC3\_ManageHotelRooms |
| 22 | 4.1 | Each room profile shall have attributes and a list of the rooms inventory. | SW | UC3\_ManageHotelRooms |
| 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 | UC4\_ManageReservations |
| 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 | UC4\_ManageReservations |
| 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 | UC4\_ManageReservations |
| 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 | UC5\_Billing |
| 34 | 4.2 | The PRMS shall allow employees to modify a reservations timespan to extend or shorten a guest’s stay. | SW | UC4\_ManageReservations |
| 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 | UC5\_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 | UC6\_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 | UC6\_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 | UC6\_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 | UC6\_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 | UC7\_ManageEventRooms |
| 42 | 5.1 | The PRMS shall provide functionality for describing the number of events rooms in the resort. | SW | UC7\_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 | UC8\_ManageBooking, UC5\_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 | UC8\_ManageBooking |
| 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 | UC9\_CreateOrder, UC10\_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 | UC9\_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 | UC9\_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 | UC12\_OutputOrders |
| 58 | 6.1 | The PRMS shall allow the employee to update the state of the order. | SW | UC10\_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 | UC12\_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 | NTH | UC12\_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 | UC9\_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 | UC9\_CreateOrder |
| 68 | 6.3 | The PRMS shall allow employees to create and modify a list of options available for catered meals. | SW | UC12\_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 | UC9\_CreateOrder |
| 71 | 6.4 | The PRMS shall provide an interface that provides functionality for creating general orders. | SW | UC9\_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 | UC9\_CreateOrder |

# Use Cases and Sequence Diagrams

Use Case 1: User Login

**Overview:** The User Login menu enables users to login their credentials upon opening the software, allowing access to the main functionalities of the software.

**Preconditions:**

1. No user is currently logged in
2. The user’s profile information is registered into the system database
3. The PRMS\_Desktop\_View is displayed

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. Enter characters into user name box | 1. Username literals are displayed |
| 1. Enter characters into password box | 1. Password field is filled with obscured characters |
| 1. User selects the “login” button | 1. The Username and user pin field is updated in the profile checker object. Credentials in profile checker are compared to Database, and when found, PRMS\_Desktop\_View enables access to other tabs of the user interface. |

**Scenario Notes:**

Items 1 and 2 can be done in any order. Thus Action item 3 successfully occurs after the procession of items 1 and 2,

**Postconditions:**

1. User will be prompted with an error message due to an incorrect login (if “Login” button was selected)
2. If the credentials are correct, access to the other tabs of the user interface will be granted (if “Login” button was selected)

|  |  |
| --- | --- |
| **Required GUI:**  PRMS\_Desktop\_View | **Use Cases Utilized:**  None |
| **Exceptions:**  In the case that the Database cannot be accessed, a resolution would be to copy the metadata into a master file. | **Timing Constraints:**  None |

**Rationale:** To allow only authorized employees to access the system

System Boundary

PRMS Desktop View

Profile Checker

Database

User enters credentials

User selects login button

checkCredentials

Credentials correct

Access is granted

Credentials incorrect

Display error message

Use Case 2: Manage Employee Profiles

**Overview:** Managerial users are able to create, modify, and remove employee profiles.

**Preconditions:**

1. The user currently logged in has the job title of Manager
2. The employee management tab is displayed

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User enters first name literal | 1. Displays new first name literal |
| 1. User enters last name literal | 1. Displays last name literal |
| 1. User selects a job title from the drop-down box | 1. Displays selected job title |
| 1. User enters a username literal | 1. Displays username literal |
| 1. User enters a password literal | 1. Password field is filled with obscured characters |
| 1. User enters a confirm password literal | 1. Confirm password field is filled with obscured characters |
| 1. User selects either “Create,” “Modify,” or “Remove” radio buttons | 1. The corresponding radio button is displayed as selected |
| 1. User selects the apply button | 1. The appropriate employee object is created, modified, or removed based on the selected radio button. |

**Scenario Notes:**

If the user selects “Create” on item 7, items 1-8 are mandatory. If the user selects “Modify” or “Remove” only items 4, 7, and 8 are mandatory. Items 1-7 may be completed in any order, but item 8 must be performed last.

**Postconditions:**

1. If “Create” was selected and all items are completed, a new employee profile will be added to the database.
2. If “Modify” was selected, the employee profile in the database with the given username will be updated with the values provided.
3. If “Remove” was selected, the employee profile in the database with the given username will be removed from the database.

|  |  |
| --- | --- |
| **Required GUI:**  PRMS\_Desktop\_View | **Use Cases Utilized:**  None |
| **Exceptions:**  If the user is creating a profile but all fields are not filled, the username is already taken, or the password fields do not match, an error message will be displayed. | **Timing Constraints:**  None |

**Rationale:** To manage employee profiles to ensure correct access and credentials

System Boundary

PRMS Desktop View

Employee Object

Database

User enters credentials

User selects create, modify, or remove buttons

Create employee if selected

Credentials correct

User selects apply button

Credentials invalid, display error

Modify employee if selected

Remove employee if selected

Credentials valid, display success

Use Case 3: Manage Hotel Rooms

**Overview:** Users are able to create, modify, and remove hotel rooms and hotel room attributes.

**Preconditions:**

1. The hotel room management tab is displayed

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User enters floor integer or integer range | 1. Displays new floor value |
| 1. User enters room number integer or integer range | 1. Displays new number value |
| 1. User enters room price | 1. Displays new price value |
| 1. User enters the number of beds in the room | 1. Displays new beds value |
| 1. User selects values for “Allows Pets,” “Allows Smoking,” and “Disability Accessible” checkboxes | 1. Displays the selected checkboxes as enabled or disabled based on the user selection |
| 1. User selects either “Create,” “Modify,” or “Remove” radio buttons | 1. The corresponding radio button is displayed as selected |
| 1. User selects the apply button | 1. The appropriate hotel object is created, modified, or removed based on the selected radio button. |

**Scenario Notes:**

For items 1 and 2, a user may enter an integer between 1 and 99, inclusive, or a range of integers in the format “x-y” where x < y and both x and y are between 1 and 99 inclusive. If a range is entered, the user may create, modify, or remove more than 1 hotel room at a time.

If the user selects “Create” on item 7, items 1-7 are mandatory. If the user selects “Modify” or “Remove” only items 1, 2, and 7 are mandatory. Items 1-6 may be completed in any order, but item 7 must be performed last.

**Postconditions:**

1. If “Create” was selected and all items are completed, a new hotel room will be added to the database.
2. If “Modify” was selected, the hotel room in the database with the given floor and number will be updated with the values provided.
3. If “Remove” was selected, the hotel room in the database with the given floor and number will be removed from the database.

|  |  |
| --- | --- |
| **Required GUI:**  PRMS\_Desktop\_View | **Use Cases Utilized:**  None |
| **Exceptions:**  If the user is creating a hotel room but all fields are not filled, the floor and number are already taken, an error message will be displayed. | **Timing Constraints:**  None |

**Rationale:** To provide flexibility in the number and type of rooms available. The software is designed to accommodate resorts of different sizes.

System Boundary

PRMS Desktop View

HotelRoom Object

Database

User enters values

User selects create, modify, or remove buttons

Create room(s) if selected

Values correct

User selects apply button

Values invalid, display error

Modify room(s) if selected

Remove room(s) if selected

Values valid, display success

Use Case 4: Create Hotel Reservation

**Overview:**

This Use Case enables the user, the employee, to make create hotel reservation functions for the customer.

**Preconditions**:

1. The user needs to reserve a hotel room for a customer.
2. The database is accessible.
3. The Reservation Room tab is displayed.

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The user clicks on text field of “Date from” and “to”, and enters the dates of the requested reservation. | 1. The two text fields of “Date from” and “to” will have numerical values representing dates. |
| 1. The user enters the price range the customer is searching for in the price range text fields | 1. The two text fields of the “Price Range” will represent numeral values representing the price range of available hotels |
| 1. The user clicks on a drop box of “Smoking” and enter a Boolean value for the hotel reservation | 1. The drop box will have a Boolean value representing whether the room is smoking or smoke free. |
| 1. The user clicks on a drop box of “Pets” and enter a Boolean value for the hotel reservation | 1. The drop box will have a Boolean value representing whether the room accommodates pets or not. |
| 1. The user clicks on a drop box of “Kitchen” and enter a Boolean value for the hotel reservation | 1. The drop box will have a Boolean value representing whether the room has a kitchen or not. |
| 1. The user clicks on a drop box of “Fridge” and enter a Boolean value for the hotel reservation | 1. The drop box will have a Boolean value representing whether the room has a fridge or not. |
| 1. The user clicks on query button to search for all of the hotel room that meets the criteria. | 1. There will be a list of available hotel room numbers that will appear on the result section. |
| 1. The user selects any one of the hotel rooms available on the list under the result section. | 1. The selected hotel room number will be highlighted in blue. |

**Scenario Notes:**

Item 1 to 6 can be done in any order, so it does not affect each other.

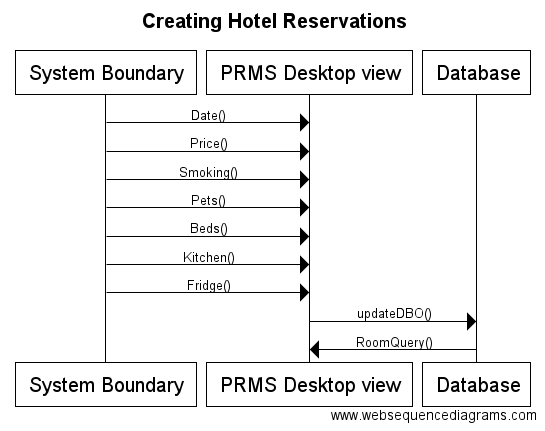
**Postconditions**:

A hotel room will be reserved for the customer based on the specification provided.

The database will update to where there will be one less room.

A page will appear to let the customer input billing information.

|  |  |
| --- | --- |
| **Required GUI:**  Mouse and keyboard | **Use Cases Utilized:**  none |
| **Exceptions:**  The DB cannot be accessed.  There are no available conference rooms to be reserved. | **Timing Constraints:**  none |



Use Case 6: Modify or Output Room Billing

**Overview:**

This Use Case enables the user, the employee, to modify or output room billing for reservations that have been placed.

**Preconditions:**

1. The user needs to select the reserved room for billing
2. The database is accessible.
3. The Reservation Room tab is displayed.

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The user will go to the payment info section, click on the text field of name and enter the name of the customer. | 1. The text field of name will have the name of the customer represented by letters. |
| 1. The user will click on the text field of the credit card number and credit card expiration date. The user will enter the information of the customer credit card. | 1. The text field of credit card number and expiration date will have credit card information of the customer in numerical value. |
| 1. The user will click on the text field of the security code. The user will enter the information of the security code from the customer credit card. | 1. The text field of the security code will have security information of the customer as a numerical value. |
| 1. The user will click on the text field of Notes. The user will enter the additional information regarding billing. | 1. The text field of Notes will have any additional information represented by letters. |
| 1. The user will click on the Output Bill button after all of the information is filled. | 1. It will send the customer to a page that tells them the billing information. |

**Scenario Notes:**

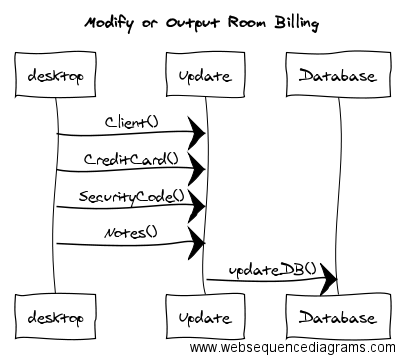
Item 1 to 4 can be done in any order, so it does not affect each other.

**Postconditions:**

1. The hotel room billing information will be saved for the customer based on the specification provided.
2. The database will update the relevant billing information.
3. A page will appear to let customer have information of their billing information room.

|  |  |
| --- | --- |
| **Required GUI:**  **Mouse and keyboard** | **Use Cases Utilized:**  **None** |
| **Exceptions:**   1. The DB cannot be accessed. 2. There are no available conference rooms to be reserved. | **Timing Constraints:**  **None** |

**Rationale:** Accurate billing is necessary for the function of any business



Use Case 6: Manage Room Maintenance

**Overview:** This Use Case enables the user, the employee, to manage hotel room maintenance for the hotel.

**Preconditions:**

1. The user needs to select a hotel room to modify.
2. The database is accessible.
3. The Maintenance tab is displayed.

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The user clicks on a room number from the list labeled “Room List”. | * The room which was clicked will be selected. * The table labeled “Room Inventory” will be populated with the selected room’s inventory items. * The Room service textfield will show the last room cleaning date * The room service “Update” button will become clickable |
| 1. The user clicks the Date Last Cleaned “Update” button. | * The new date will be updated in the database. * The new date will be updated in the textfield. * A “date updated!” label will appear. |
| 1. With a room already selected, the user clicks on an item in the “Room Inventory” list. | * The item which was clicked will be selected. |
| 1. With an inventory item selected, the user clicks the “Replenish” button | * The selected inventory item’s quantity will be set to match expected quantity both in the GUI and in the database. |
| 1. The user clicks any of the Room Inventory headers | * The Room Inventory Table will be sorted according to the header clicked. |

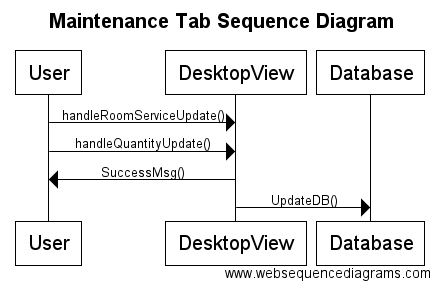
**Scenario Notes:**

To initiate item 2, only a room has to be selected, but for 3, 4 and 5, an inventory item must also be selected.

**Postconditions:**

1. The hotel maintenance updates will be saved in the database based on the specification provided.

|  |  |
| --- | --- |
| **Required GUI:**  **Mouse and keyboard** | **Use Cases Utilized:**  **none** |
| **Exceptions:**   1. The DB cannot be accessed. | **Timing Constraints:**  **none** |

****

Use Case 7: Manage Event Rooms

**Overview:** Managers can reserve available an event room for a certain amount of time at a certain cost.

**Preconditions:**

1. The PRMS\_Desktop\_View is displayed
2. Manager user is logged in into system
3. The user has clicked on the event booking tab in the interface

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User selects the name of the room from the drop box. | 1. The event room will be updated in the database and the field will be occupied with characters and numerical. |
| 1. User selects the price of the room from the drop box. | 1. The price of the room will be selected and the field will be occupied by numerical. |
| 1. User selects the capacity of the room from the drop box. | 1. The capacity of the room will be selected and updated. The field will be occupied by integers. |
| 1. User selects the dates that the event room will be occupied. | 1. The date that the event room will be booked will be updated and selected. The field will be occupied by integers. |
| 1. User enter all of the text field of the billing information | 1. The text fields with billing information will be filled with characters and numerical. |
| 1. User clicks the “Create Booking”. | 1. The event room with the criteria established will be stored in the database and booked. |

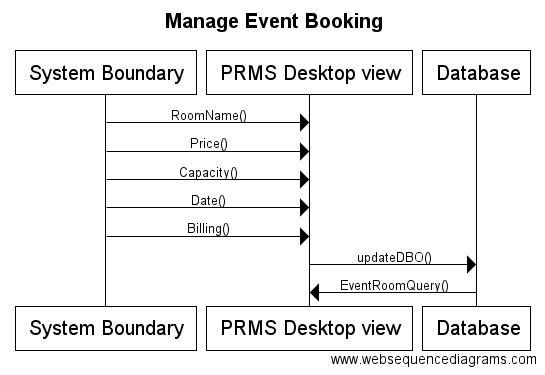
**Scenario Notes:**

Items 1,2,3,4, and 5 can be done in any order, as long as all action items are initiated in entirety before proceeding to item 6.

**Postconditions:**

1. Database is updated (if “Create Booking” button was selected)
2. The M/A is returned to the PRMS\_Desktop\_View.

|  |  |
| --- | --- |
| **Required GUI:**  PRMS\_Desktop\_View | **Use Cases Utilized:**  none |
| **Exceptions:**  In the case that the Database cannot be accessed, a resolution would be to copy the metadata into a master file | **Timing Constraints:**  none |

****

Use Case 8: Manage Event Booking

**Overview:** This Use Case enables the user, the employee, to make book a room for an event, requested by the customer, for a certain time and day.

**Preconditions:**

1. The user needs to reserve an event room for a customer.
2. The database is accessible.
3. The event booking tab is displayed.

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User selects the name of the room from the drop box. | 1. The event room will be updated in the database and the field will be occupied with characters and numerical. |
| 1. User selects the price of the room from the drop box. | 1. The price of the room will be selected and the field will be occupied by numerical. |
| 1. User selects the capacity of the room from the drop box. | 1. The capacity of the room will be selected and updated. The field will be occupied by integers. |
| 1. User selects the dates that the event room will be occupied from drop box. | 1. The date that the event room will be booked will be updated and selected. The field will be occupied by integers. |
| 1. User enter all of the text field of the billing information | 1. The text fields with billing information will be filled with characters and numerical. |
| 1. User clicks the “Create Booking”. | 1. The event room with the criteria established will be stored in the database and booked. |

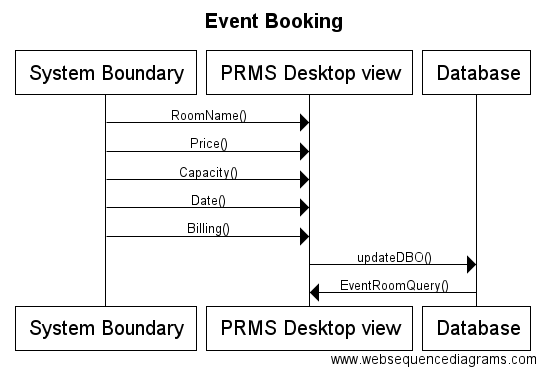
**Scenario Notes:**

Item 1 to 4 can be done in any order, so it does not affect each other.

**Postconditions:**

1. An event room will be reserved for the customer based on the specification provided.
2. The database will update to where there will be one less room.
3. A page will appear to let customer have information of their reserved event room.

|  |  |
| --- | --- |
| **Required GUI:**  **Mouse and keyboard** | **Use Cases Utilized:**  **none** |
| **Exceptions:**   1. The DB cannot be accessed. 2. There are no available conference rooms to be reserved. | **Timing Constraints:**  **none** |

****

Use Case 9: Create Order

**Overview:** The use case is established such that actors can request and order various services through the end-users. The end-users, hotel employees, respectively work with an ordering system functionality that allows them to document the requests and services in one space

**Preconditions:**

1. The “Create Order” button is not selectable until the expected parameters have been filled
2. The user is logged into the system
3. The user’s profile information is registered into the system database
4. The PRMS\_Desktop\_View is displayed

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. User enters name of room that requested service | 1. New Room order object created. Room order field is updated |
| 1. Users select option from the “order type” dropdown menu | 1. Order type field is updated |
| 1. Users select option from the “order state” dropdown menu | 1. Order state field is updated |
| 1. User inputs values for date and time | 1. Date field is updated |
| 1. User selects whether order is a reservation or booking for billing data | 1. Billing type field is updated |
| 1. User enters the invoice numbers for billing. | 1. The invoice field will be filled with numerical. |
| 1. User selects the “Create Order” button | 1. New order object added to database. Success message displayed |

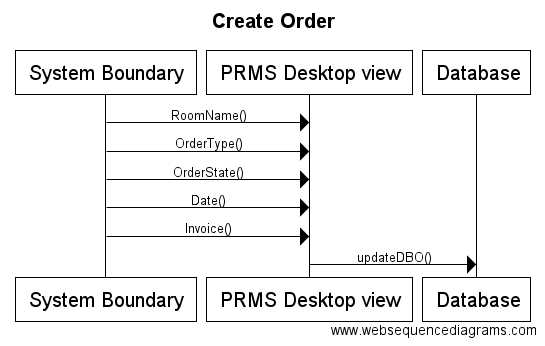
**Scenario Notes:**

Action items 1,2,3,4, and 5 can be done in any order, as long as all action items are done completely before proceeding to action item 6.

**Postconditions:**

1. Room order field is updated(if “Create order” button was selected)
2. Order type field is updated(if “Create order” button was selected)
3. Order state field is updated (if “Create order” button was selected)
4. Time field is updated(if “Create order” button was selected)
5. Billing type field is updated (if “Create order” button was selected)
6. Order request object is added to database(if “Create order” button was selected)
7. Order\_Made\_Pop-up is created(if “Create Order” button was created)

|  |  |
| --- | --- |
| **Required GUI:**  PRMS\_Desktop\_View | **Use Cases Utilized:**  None |
| **Exceptions:**  None | **Timing Constraints:**  None |

****

Use Case 10: Modify or Update Order

**Overview:** The user is able to look up and manage orders.

**Preconditions:**   
1. The user has to be logged in as manager.

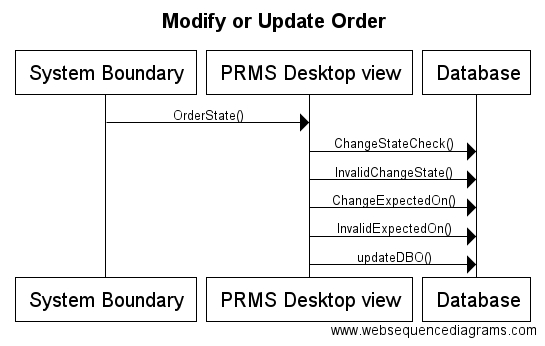
**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| User clicks the Manage Orders tab in the Orders page | A “change state” dropdown menu ,an “change expected date” field, and an inactive “submit changes” button appear |
| Scenario: User changes state |  |
| User clicks on the relevant order in the Order List | Order becomes selected |
| User clicks on a new state from the “change state” dropdown menu: “ordered”, “preparing”, or “completed” | New state becomes selected |
| User chooses “completed” | The “change expected date” field becomes mandatory and now reads, “delivered on” |
| User fills the “delivered on” field | The date is checked. If User provided a date not later than today, “Submit changes” button becomes active. If the User provided a date later than today, an error message appears. |
| User chooses “ordered” or “preparing” | The “submit changes” button becomes active. |
| User clicks “submit changes” | The new state of the selected item becomes updated in the database, and a success message appears. |
| Scenario: User changes expected date |  |
| User clicks on the relevant order in the Order List | Order becomes selected |
| User changes the date within the “change expected date” field and clicks “submit changes” | The date is checked. If User provided a date not earlier than today, “Submit changes” button becomes active. If the User provided a date earlier than today, an error message appears. |
| User clicks "submit changes” | The new expected date of the selected item becomes updated in the database, and a success message appears. |

**Scenario Notes:** The User can change both state and “expected on” date before clicking “submit changes” instead of doing it one by one.

**Postconditions:** If changes are made, they are applied to the database.

|  |  |
| --- | --- |
| **Required GUI:**  **Keyboard and Mouse** | **Use Cases Utilized:**  **None** |
| **Exceptions:**  **None** | **Timing Constraints:**  **None** |



Use Case 11: Output Orders

**Overview:** This Use Case enables the users or employees to output orders.

**Preconditions:**

1.The user needs to have an order that already have orders created and modified.

**Scenario**

|  |  |
| --- | --- |
| Action | Software Reaction |
| 1. The user clicks on the on the Output bottom after the modifying section of the order. | 1. The order will be in a pop up window. |

**Scenario Notes:**

Item 1 can only be performed when an order is created or modified.

**Postconditions:**

The itemized order will be in a pop up window and sent to customers.

|  |  |
| --- | --- |
| **Required GUI:**  **Mouse and keyboard** | **Use Cases Utilized:**  **None** |
| **Exceptions:**  **None** | **Timing Constraints:**  **None** |

Use Case 12: Manage Restaurant Menu

**Overview:** TheUser is able to add, remove, or edit items on the restaurant menu and the catered meals menu.

**Preconditions:**   
1. User has logged in as manager.

2. For adding an item, precondition is: no respective restaurant or catered meals menu item is selected.

3. For editing an item, precondition is: a respective restaurant or catered meals menu item is selected.

4. For removing an item, precondition is: a respective restaurant or catered meals menu item is selected.

**Scenario**

|  |  |
| --- | --- |
| **Action** | **Software Reaction** |
| Scenario: Add Menu Item |  |
| User clicks the deselect Under the Restaurant menu section (optional if nothing if no restaurant menu item is selected beforehand) | The text fields are in Add Item mode  The “AddEdit” button is in Add mode. |
| User fills in the item name [required], price [required], and description [optional] fields, and clicks the “Add Item” button | A popup dialog appears asking the user if they are sure they want to add this item. |
| User Clicks OK in the popup dialog | The information is saved in the database and and the GUI table is updated accordingly. |
| User clicks Cancel in the dialog popup | The changes are not applied to the database. |
| User enters name but doesn’t enter cost, or vice versa | No changes to the database are applied and an error message appears. |
| User doesn’t enter neither name nor cost | No changes to the database are applied and an error message appears. |
| User provides a non-numerical value for price | No changes to the database are applied and an error message appears. |
| Scenario: Remove Item |  |
| User clicks on an existing menu item | The item becomes selected.  The “AddEdit" button is in Edit mode.  The “delete” and “deselect” buttons become active. |
| With an item selected, the user clicks the “Delete Selected Item” button | A Confirmation dialog popup appears. |
| User clicks OK in the dialog popup | The relevant item is removed from the database and a success message is displayed.  Item is deselected and the “delete” and “deselect” buttons become inactive. |
| User clicks Cancel in the dialog popup | The changes are not applied to the database. |
| Scenario: Edit Item |  |
| User selects an item they wish to edit | The "AddEdit" button is in Edit mode  The name, cost, and description fields become editable for that item. |
| The user makes the relevant changes and clicks the “save” button | A Confirmation dialog popup appears. |
| User clicks OK in the dialog popup | The changes are applied to the item in the database and a success message is displayed. |
| User clicks Cancel in the dialog popup | The changes are not applied to the item in the database. |
| User provides a non-numerical value for price | No changes to the database are applied and an error message appears. |

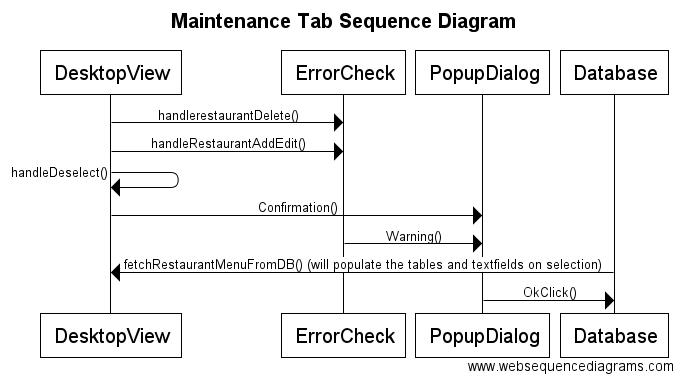
**Scenario Notes:**

The page contains two tables for which the instructions and use cases are identical. However, one table is responsible for restaurant menu items, while the other is for catered meal menu items.

If no item is selected, clicking the “save” or “delete” buttons will have no effect - they remain greyed out and disabled.

**Postconditions:** All relevant modifications, if any, are applied to the database.

|  |  |
| --- | --- |
| **Required GUI:**  **Keyboard and Mouse** | **Use Cases Utilized:**  **None** |
| **Exceptions:**  **None** | **Timing Constraints:**  **None** |



# Category Interaction Diagram

Interface\_CAT

Processing\_CAT

Objects\_CAT

Database\_CAT

# Object Design

public class BillableItem {

private SimpleStringProperty billableName = new SimpleStringProperty();

private SimpleDoubleProperty price = new SimpleDoubleProperty();

private SimpleStringProperty time = new SimpleStringProperty();

private SimpleStringProperty invoiceUID = new SimpleStringProperty();

private SimpleStringProperty billableUID = new SimpleStringProperty();

public BillableItem(String billableName, double price, String time, String invoiceUID, String billableUID) {

setBillableName(billableName);

setPrice(price);

setTime(time);

setInvoiceUID(invoiceUID);

setBillableUID(billableUID);

}

// Include getters and setters

}

public class CateredMealItem {

private String mealName;

private double pricePerSeat;

private String mealDescription;

public CateredMealItem(String mealName, double pricePerSeat, String mealDescription) {

this.mealName = mealName;

this.pricePerSeat = pricePerSeat;

this.mealDescription = mealDescription;

}

// Include getters and setters

}

public class Employee {

private SimpleStringProperty firstName = new SimpleStringProperty("");

private SimpleStringProperty lastName = new SimpleStringProperty("");

private SimpleStringProperty jobTitle = new SimpleStringProperty("");

private SimpleStringProperty username = new SimpleStringProperty("");

private String password;

public Employee( String first, String last, String job, String user, String pass ) {

setFirstName(first);

setLastName(last);

setJobTitle(job);

setUsername(user);

setPassword(pass);

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the employee

}

public class EventRoom {

private SimpleStringProperty roomName = new SimpleStringProperty("");

private SimpleDoubleProperty price = new SimpleDoubleProperty();

private SimpleIntegerProperty maxCapacity = new SimpleIntegerProperty();

private SimpleBooleanProperty hasStage = new SimpleBooleanProperty();

private SimpleBooleanProperty hasAudioVisual = new SimpleBooleanProperty();

public EventRoom(String roomName, double price, int maxCapacity, Boolean hasStage, Boolean hasAudioVisual) {

setRoomName(roomName);

setPrice(price);

setMaxCapacity(maxCapacity);

setHasStage(hasStage);

setHasAudioVisual(hasAudioVisual);

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the event room

}

public class HotelReservation {

private SimpleStringProperty roomNumber = new SimpleStringProperty();

private SimpleStringProperty startDate = new SimpleStringProperty();

private SimpleStringProperty endDate = new SimpleStringProperty();

private SimpleIntegerProperty children = new SimpleIntegerProperty();

private SimpleIntegerProperty adults = new SimpleIntegerProperty();

private Invoice bill;

public HotelReservation(String roomNumber, int adults, int children, String startDate, String endDate) {

setRoomNumber(roomNumber);

setRoomAdults(adults);

setRoomChildren(children);

setStartDate(startDate);

setEndDate(endDate);

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the reservation

}

public class HotelRoom {

private SimpleStringProperty roomNumber = new SimpleStringProperty("");

private SimpleDoubleProperty price = new SimpleDoubleProperty();

private SimpleIntegerProperty beds = new SimpleIntegerProperty();

private SimpleBooleanProperty allowsPets = new SimpleBooleanProperty();

private SimpleBooleanProperty disabilityAccessible = new SimpleBooleanProperty();

private SimpleBooleanProperty allowsSmoking = new SimpleBooleanProperty();

private SimpleStringProperty dateLastCleaned = new SimpleStringProperty();

public ArrayList<InventoryItem> inventory;

public HotelRoom(String roomNumber, double price, int beds, Boolean allowsPets, Boolean disabilityAccessible, Boolean allowsSmoking, String dateLastCleaned) {

setRoomNumber(roomNumber);

setPrice(price);

setBeds(beds);

setAllowsPets(allowsPets);

setDisabilityAccessible(disabilityAccessible);

setAllowsSmoking(allowsSmoking);

setDateLastCleaned(dateLastCleaned);

inventory = new ArrayList<InventoryItem>();

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the room

}

public class InventoryItem {

public SimpleStringProperty name = new SimpleStringProperty("");

public SimpleIntegerProperty quantity = new SimpleIntegerProperty();

public SimpleIntegerProperty expectedQuantity = new SimpleIntegerProperty();

public SimpleBooleanProperty isConsumable = new SimpleBooleanProperty();

/\* Creates an InventoryItem object

\* to be sent to the database.

\*

\* @precondition:

\* all relevant InventoryItem

\* attributes like Name, Quantity,

\* etc. have to be provided as parameters.

\*

\* @postcondition:

\* InventoryItem object is created and sent to

\* the UsedInventory.

\*/

public InventoryItem(String Name, int Quantity, int ExpectedQuantity, boolean IsConsumable) {

setName(Name);

setQuantity(Quantity);

setExpectedQuantity(ExpectedQuantity);

setIsConsumable(IsConsumable);

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the item

}

public class Invoice {

private SimpleStringProperty invoiceUID;

private SimpleStringProperty customerName;

private SimpleStringProperty creditCardNum;

private SimpleStringProperty creditCardExp;

private SimpleDoubleProperty amountPaid;

private ArrayList<BillableItem> billableItems;

public Invoice(String invoiceUID, String customerName, String creditCardNum, String creditCardExp) {

setInvoiceUID(invoiceUID);

setCustomerName(customerName);

setCreditCardNum(creditCardNum);

setCreditCardExp(creditCardExp);

setAmountPaid( 0 );

}

public ArrayList<BillableItem> getBillableItems() {

return billableItems;

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the invoice

}

public class RestaurantItem {

private SimpleStringProperty itemName = new SimpleStringProperty();

private SimpleDoubleProperty price = new SimpleDoubleProperty();

private SimpleStringProperty description = new SimpleStringProperty();

public RestaurantItem(String itemName, double price, String description) {

setItemName(itemName);

setPrice(price);

setDescription(description);

}

// Include getters and setters

@Override

public String toString() {} // Creates a plain text output of the invoice

}

# 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. |

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **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. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Event Room Booking |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will enter the room name, price of the room, capacity, the date the room will be occupied, and billing information. |
| Oracle | It is expected that the user will be able to reserve or book a room following the criteria requested by the customer. |
| Log | It was a success. An event room has been reserved or booked with the requested criteria by the user. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Create Reservation |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will enter the information required to reserve a hotel room. The user will enter the date that the room will be reserved, and get the price range of the hotel room. The user will have to also determine whether there should be a smoking area, pets allowed, a kitchen and a fridge. The user will then find available rooms of those criteria. |
| Oracle | It is expected that the user will able to successfully get a hotel room reserved with the given criteria requested by the customer. |
| Log | The user is successfully able to reserve a room for the specific criteria that the customer requested. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Modifying Reservations |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will change the criteria that were already established to another result. For example, the user will change the room criteria from allowing dogs to disallowing dogs. |
| Oracle | It is expected that the requested modifications by the customer to be successfully executed. |
| Log | It was a success. The user was able to modify the hotel reservation to a different requirement requested by a customer. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Create Order |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will enter the name of the room in the text field and then he/she will enter order type, and order state. The user will also enter the date and time in the text fields designated for it. The user selects the type of order( reservation or booking for billing). The user will click the button to create the order. |
| Oracle | It is expected that the order will be placed and recorded. |
| Log | It was a success. A new order was created and recorded to the database. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Modify Order |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will change the state of the order and the expected date and time of the order. |
| Oracle | It is expected that the modification of the order will be successful and changed to the requested modification. |
| Log | The editing of the information of the order was a success. The requested changes were made. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Output Order |
| Tester | Andrew Truong |
| Input | The user (manager/supervisor) will click on a button that would output the order. |
| Oracle | It is expected that the order will be sent to the customer to let the customers know if the order was modified or created. |
| Log | The order was successfully sent to the customer. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Create Menu Item |
| Tester | Deividas Rutkauskas |
| Input | The user clicks “deselect” to make sure the program is in “add” mode. Then the user fills in the three blanks representing name, price and description. Then, the user clicks the “Add Item” button. A popup dialog appears asking for confirmation. The user clicks “OK”. |
| Oracle | If a value was provided for name and price, and if price is a numerical value, the output is a successful update in the database which is immediately reflected in the rest aurant item menu table in the GUI. |
| Log | It was a success. A new item was created, recorded to the database, and shown in the restaurant table GUI element. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Edit Menu Item |
| Tester | Deividas Rutkauskas |
| Input | The user selects an item they want to edit. The name, price and description fields become populated with the selected item’s data, and the “deselect” and “delete” buttons become active. The user edits any combination of fields before clicking “Edit Item”. A popup dialog appears asking for confirmation. The user clicks “OK”. |
| Oracle | It is expected that the modification of the order will be successful and changed to the requested modification. |
| Log | The editing of the information of the item was a success. The selected item’s values are changed both in the database and the GUI table. |

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Name | Remove Item |
| Tester | Deividas Rutkauskas |
| Input | The user selects an item they want to edit. The name, price and description fields become populated with the selected item’s data, and the “deselect” and “delete” buttons become active. The user clicks the “delete” button. A popup dialog appears asking for confirmation. The user clicks “OK”. |
| Oracle | It is expected that the item will be removed from the database and from the GUI menu table |
| Log | The order was successfully removed from the database and from the GUI menu table. |

**Note:** The test cases were identical for both the Restaurant Menu table and the Catered Meal Menu table. The code was recycled and behavior was the same, except for the difference in the Database queries to different tables, and the different associated object initializations. However, practically the two tables can be treated as the same structure.

Test Case Rationale

In order to sufficiently test the current build of our software, we decided to use a vertical integration testing strategy. 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.

# Project Rationale

In implementing this project, the most important choices we made involved which technologies we chose to use and how our objects should be designed.

Since the course required us to develop the software in the Java language, that need not be considered. However, we had an open choice as to how we would implement data persistence (information stored, even when the software is not running) and how to create the user interface for our software.

For data persistence, we looked a variety of options, including storing the data in flat files and several different database management systems. We wanted a solution that would easily integrate in to our Java code and be relatively easy to set up. We found that SQLite fit our requirements, however there was a considerable amount of effort spent making sure every member of the team was able to write effective code.

For our user interface, we decided early that we would want a graphical user interface for our software. The preferred method for developing a GUI in Java is using a technology called JavaFX, so it was the obvious choice. It allowed those of us who were comfortable creating a user interface from scratch to directly create the interface in code, while other who were more comfortable with a drag and drop system could use a tool called SceneBuilder to create the user interface.

As for object design, we struggled with identifying the most effective way to handle connecting various objects in our database. Over the course of the project, we became more familiar with SQLite, and we were able to create a system by which an object could be linked to objects of other types using unique IDs.

# Function Point Cost Analysis

Weighting Factor Estimation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Measurement parameter** | **count** |  | **average** |  |  |
| Number of user inputs | 38 | × | 4 | = | 152 |
| Number of user outputs | 14 | × | 5 | = | 70 |
| Number of user inquires | 10 | × | 4 | = | 40 |
| Number of files | 1 | × | 10 | = | 10 |
| Number of external interfaces | 1 | × | 7 | = | 7 |
|  |  |  |  | **Total =** | **279** |

Complexity Adjustment Factor

|  |  |
| --- | --- |
| **Factor** | **Value** |
| Backup & recovery | 2 |
| Data communications | 2 |
| Distributed processing | 0 |
| Performance critical | 2 |
| Existing operating environment | 5 |
| Online data entry | 5 |
| Input transaction over multiple screens | 4 |
| Master files updated online | 3 |
| Information domain values complex | 2 |
| Internal processing complex | 1 |
| Code designed for reuse | 3 |
| Conversion/installation in design | 0 |
| Multiple installations | 5 |
| Application designed for change | 5 |
| Complexity adjustment factor | 1.17 |
| **Total** | **40.17** |

Function Point Calculation

**FP =** 279 × (0.65 + 0.01 × 40.17) = 279 × 1.0517 = 293.424

At 6.5 FP per month and $8000.00 per month, the total cost of the software would be

( 293 / 6.5 ) \* 8000 = **$360,615**

# 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**

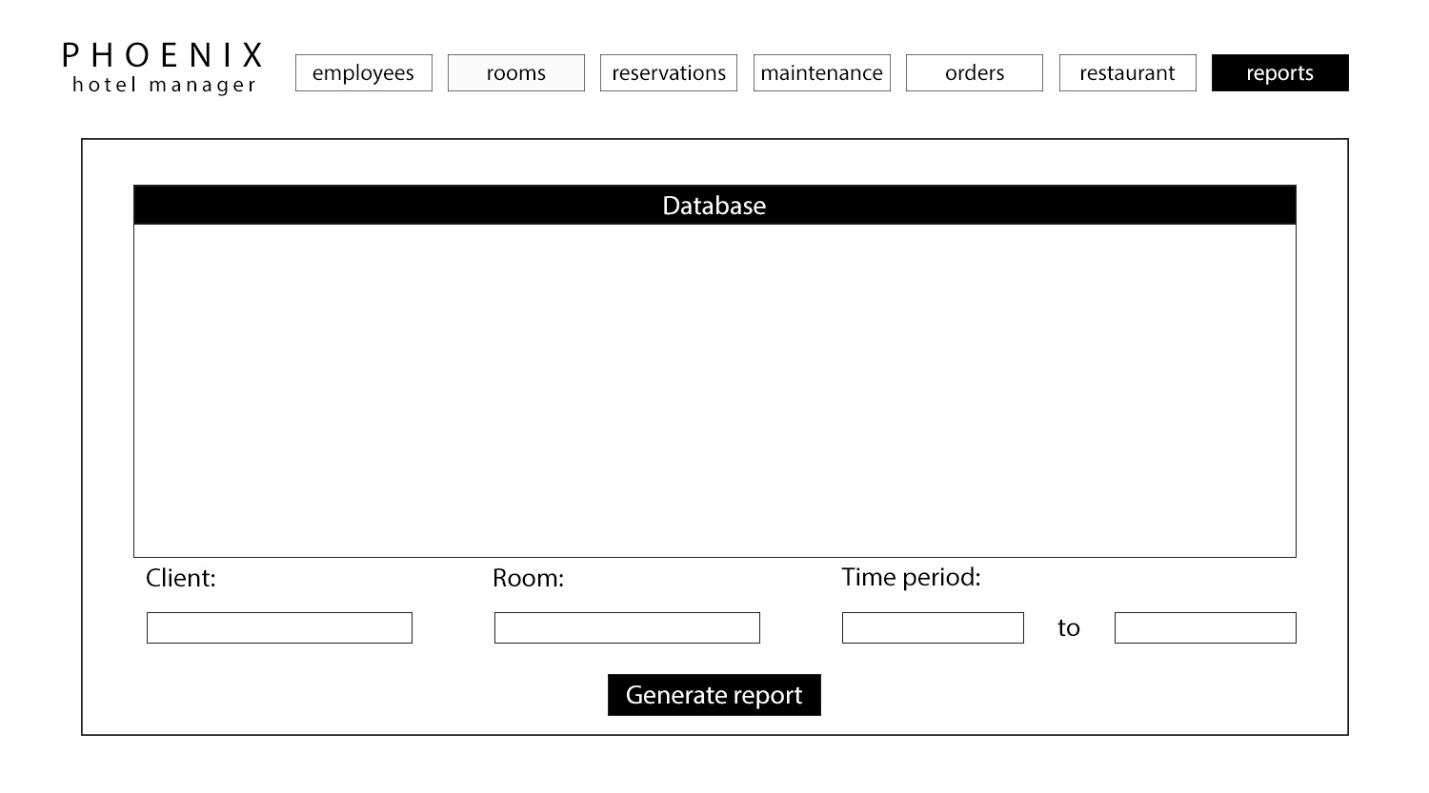
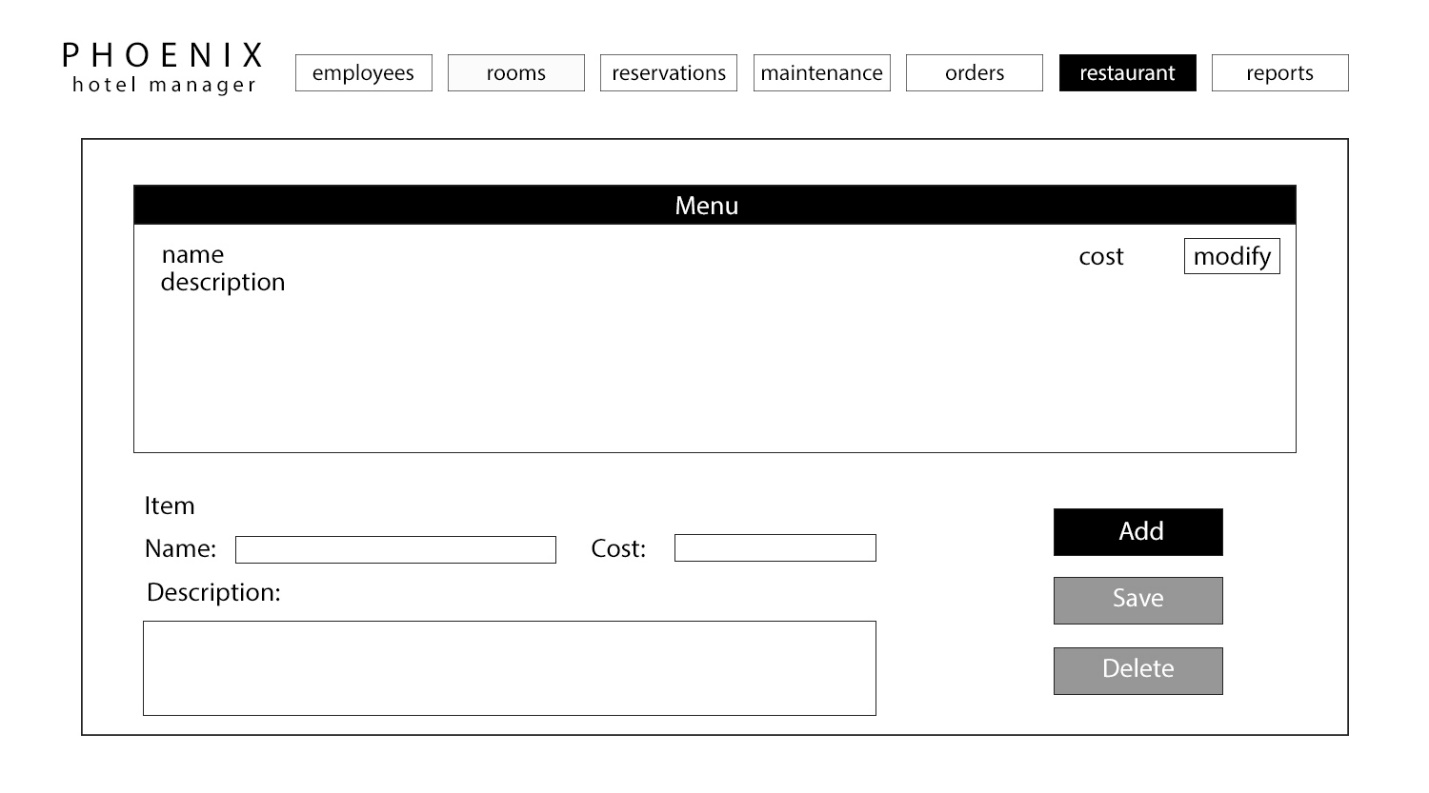
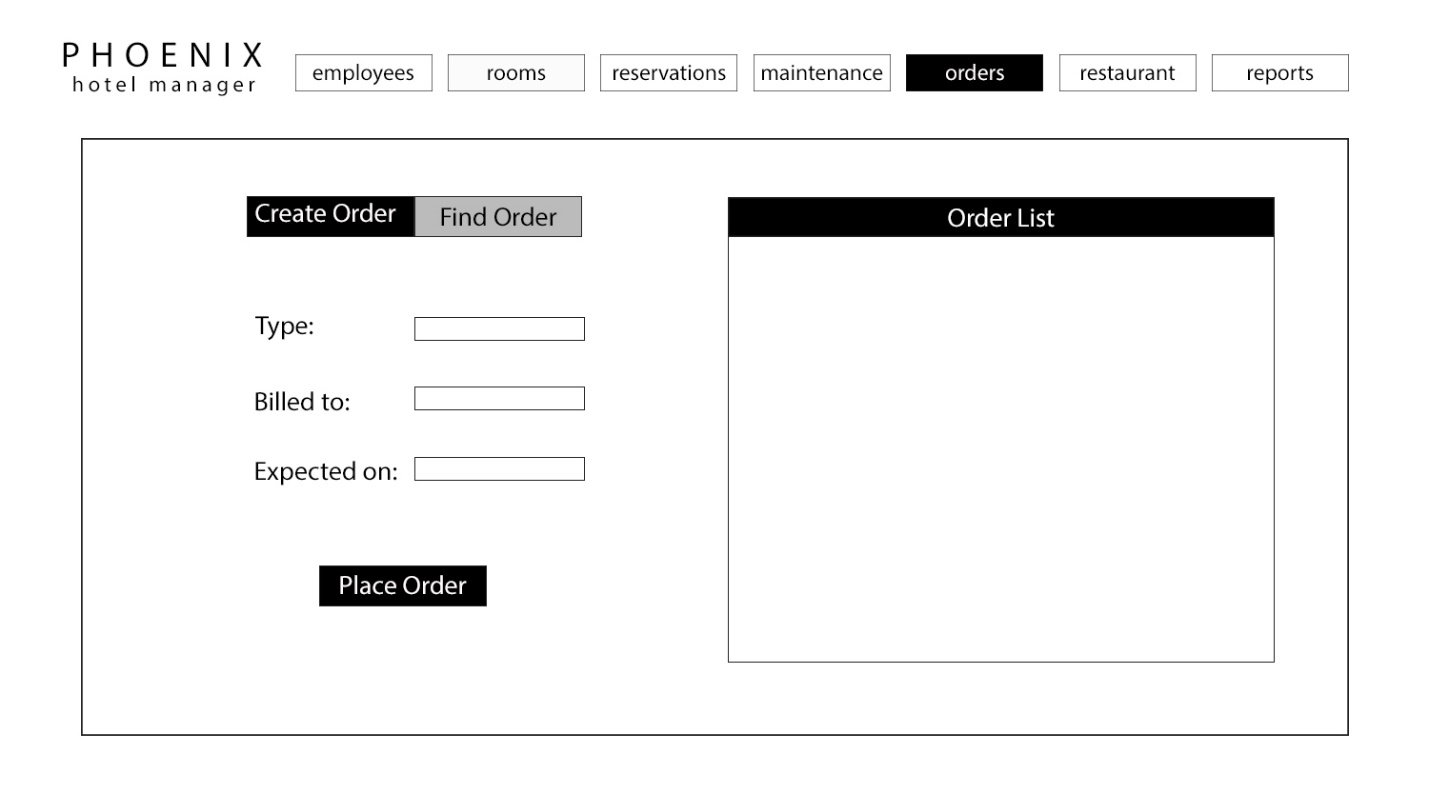
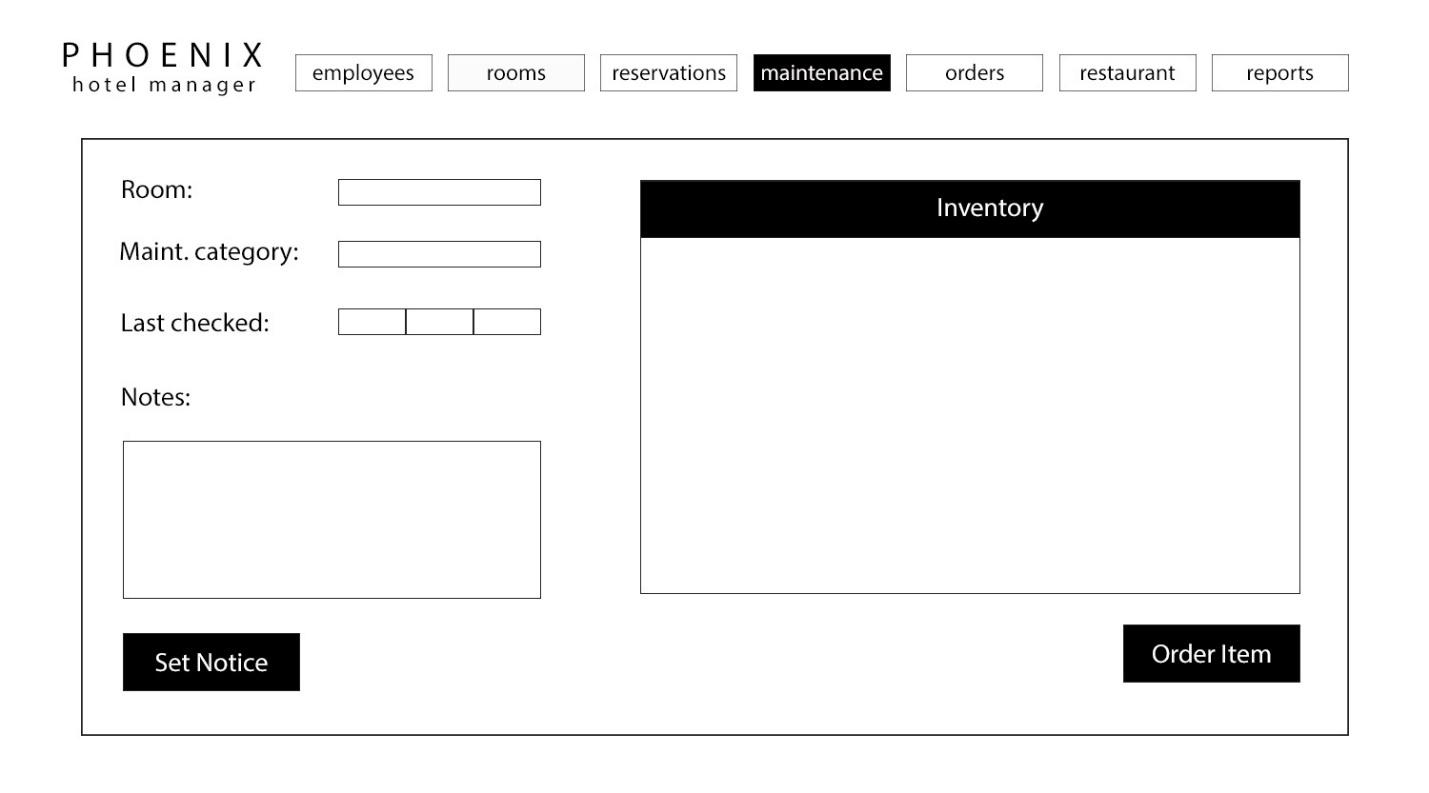
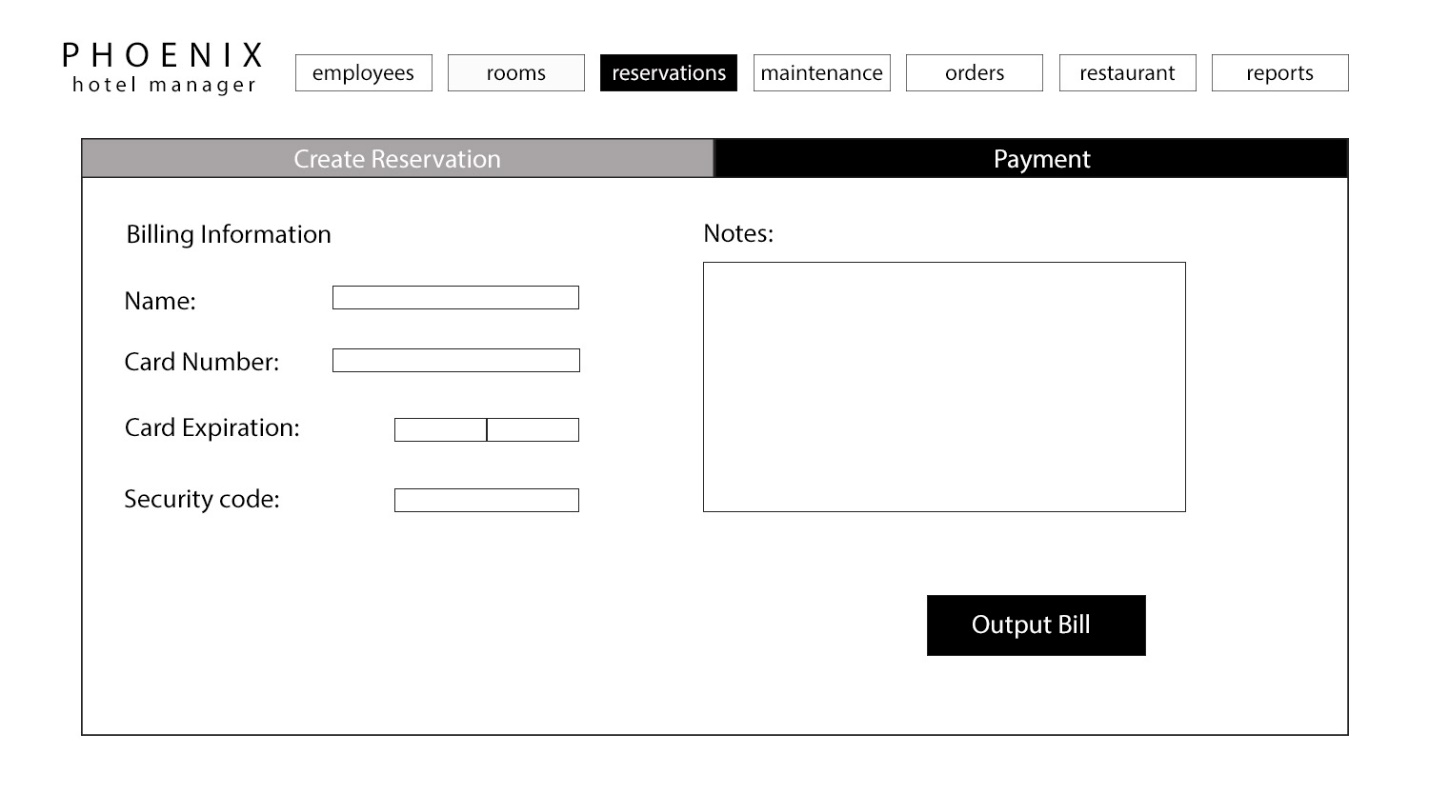
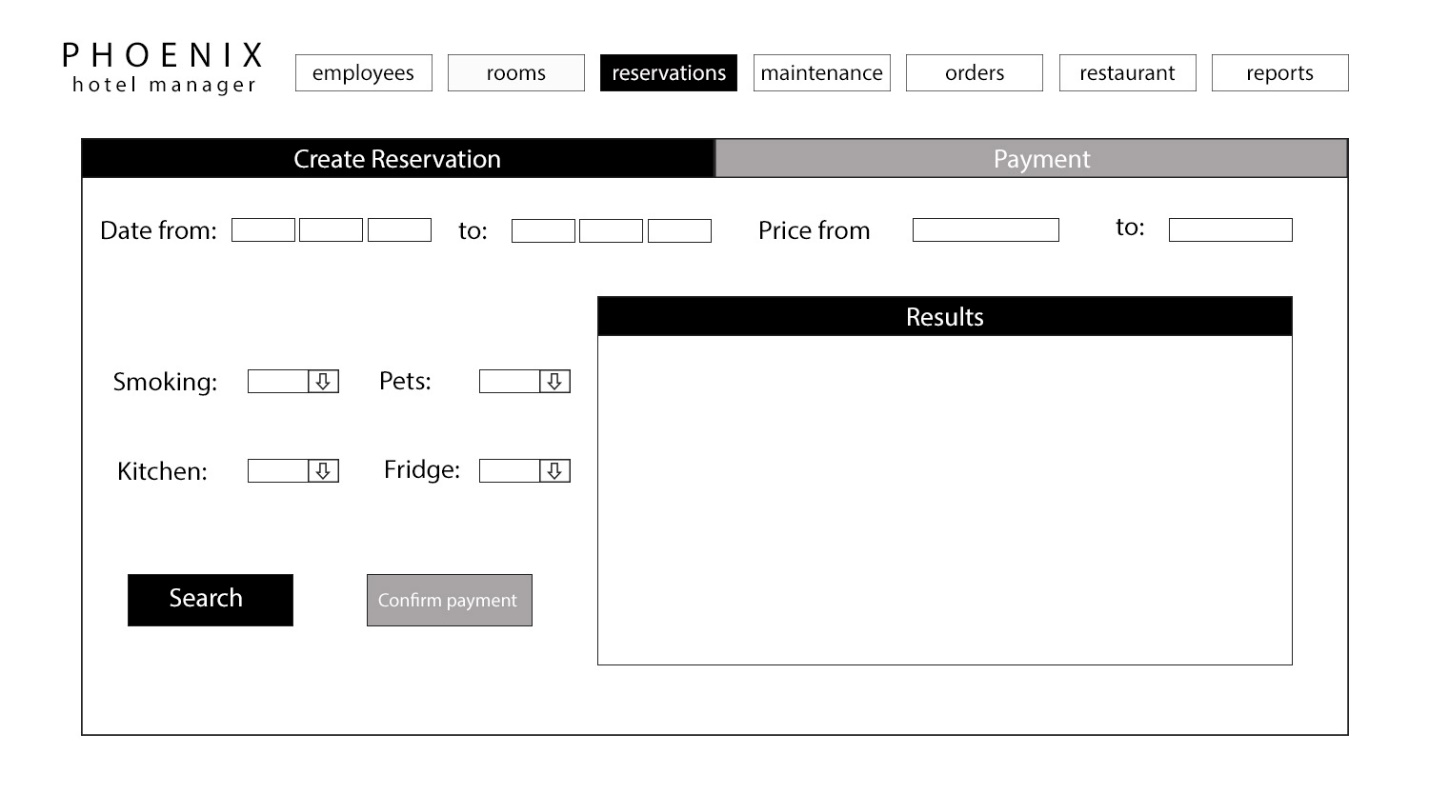
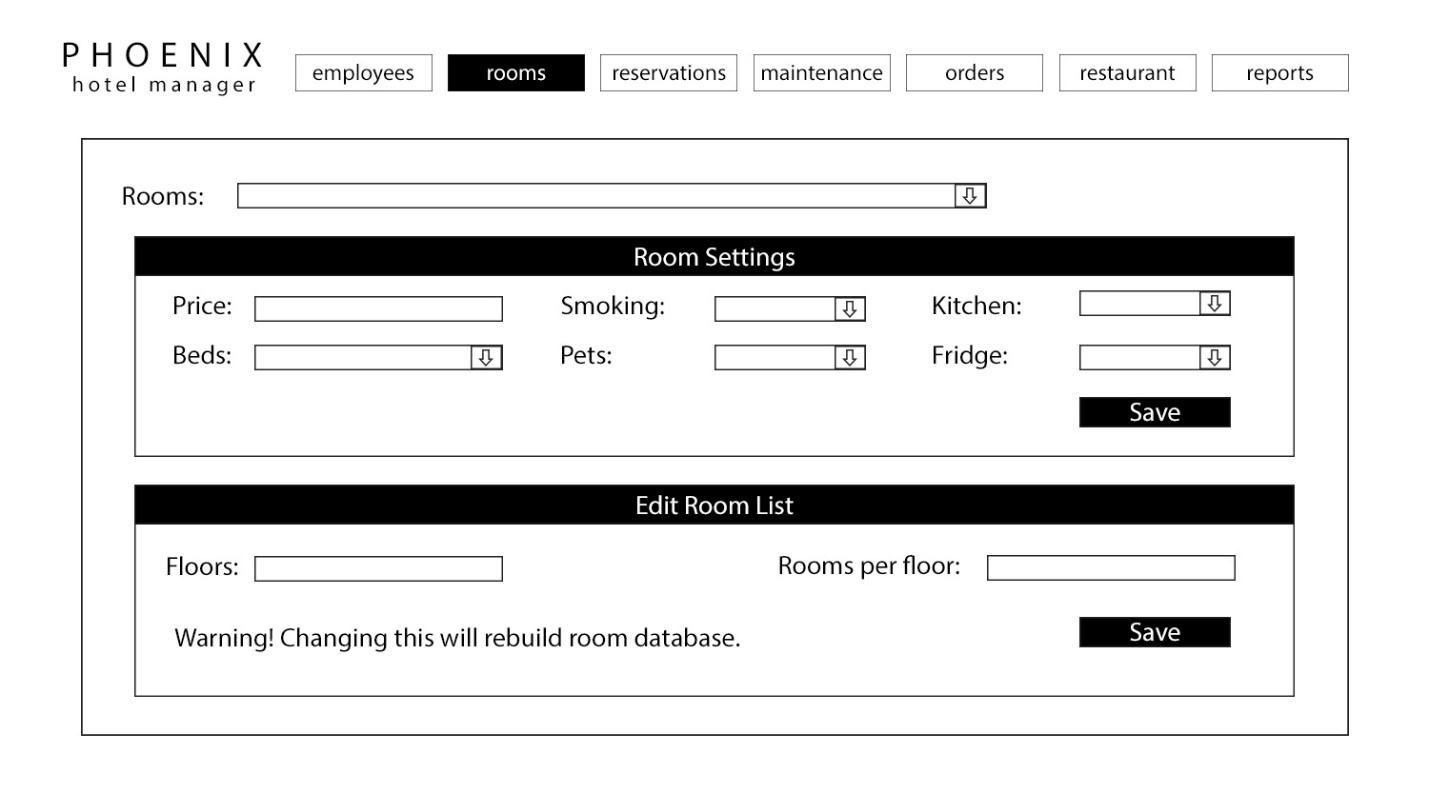
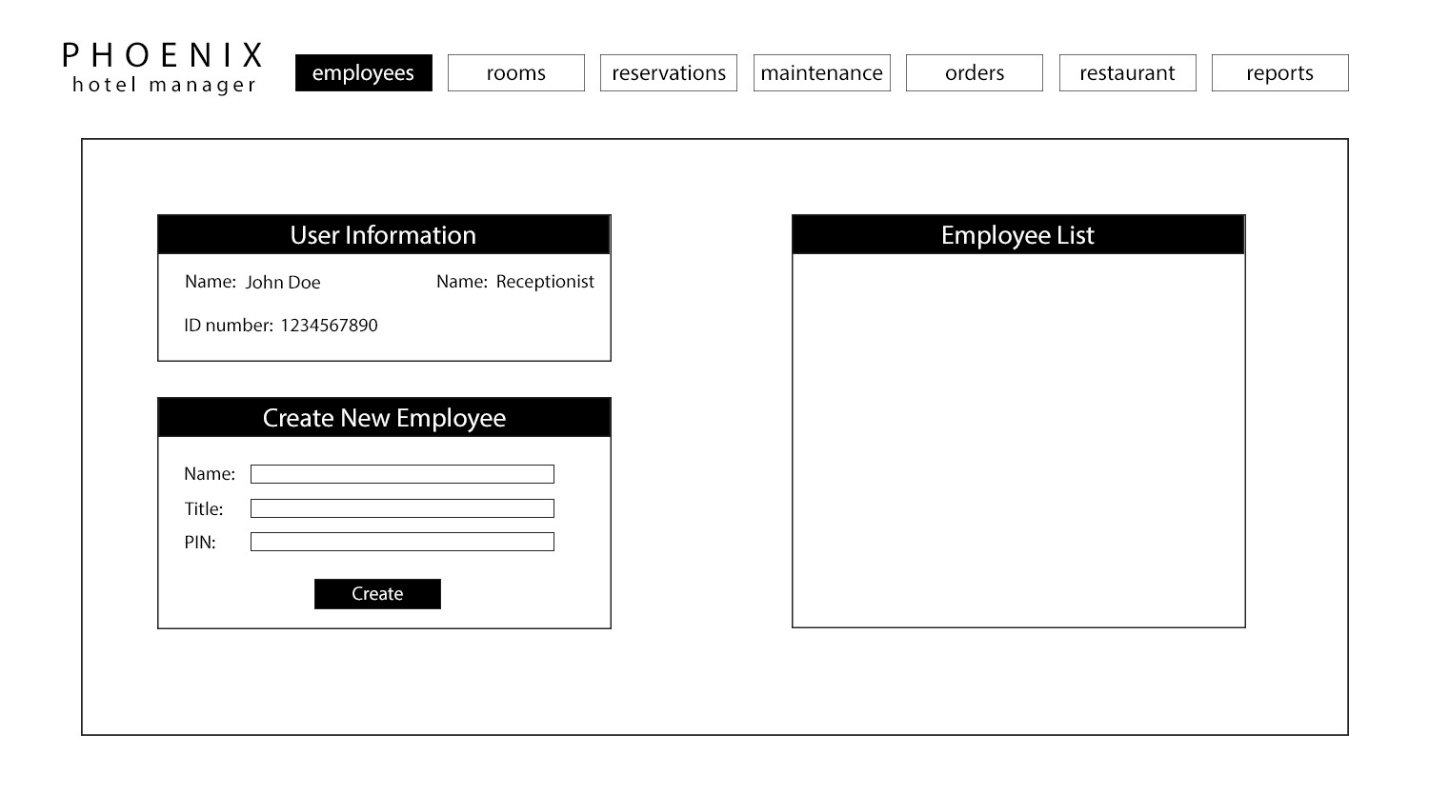
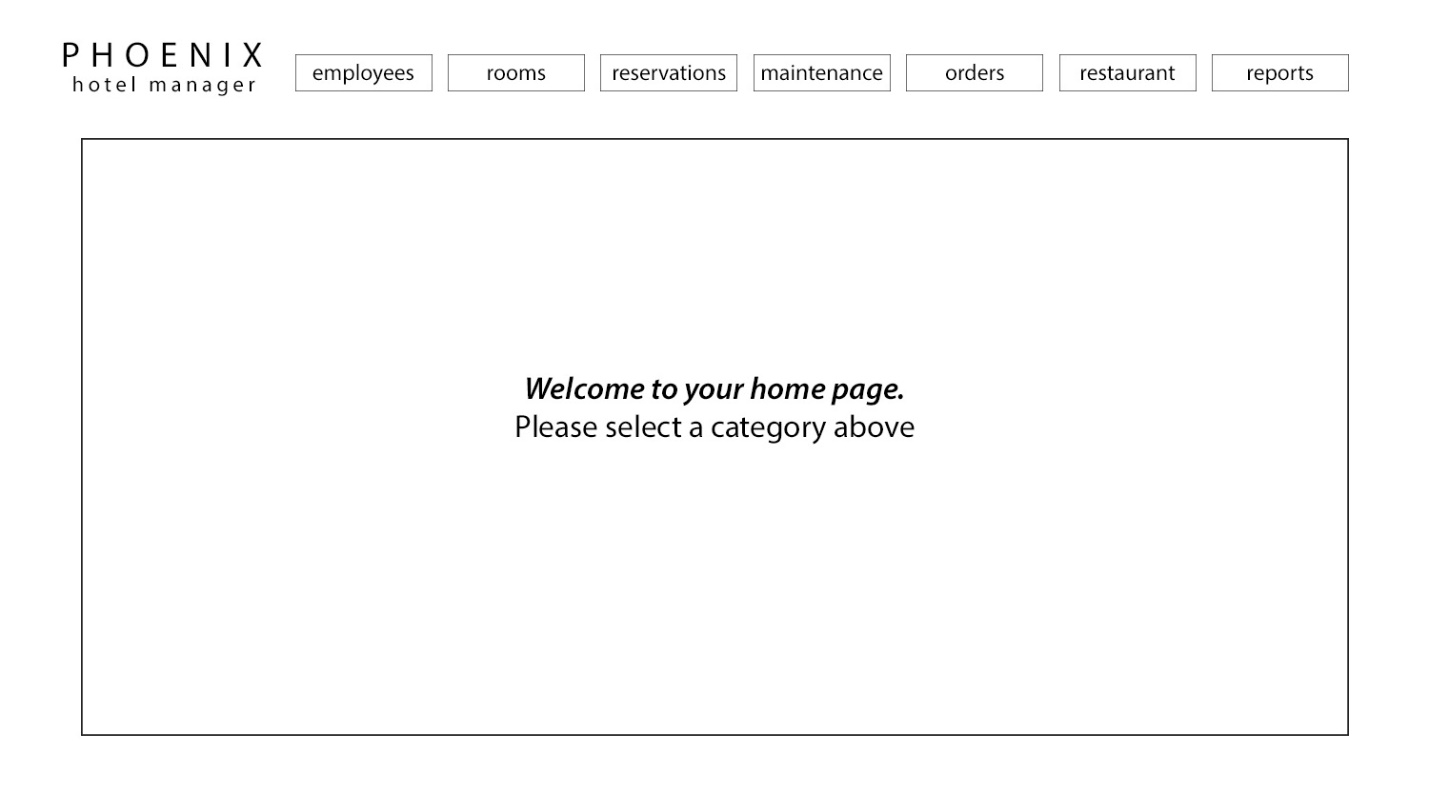
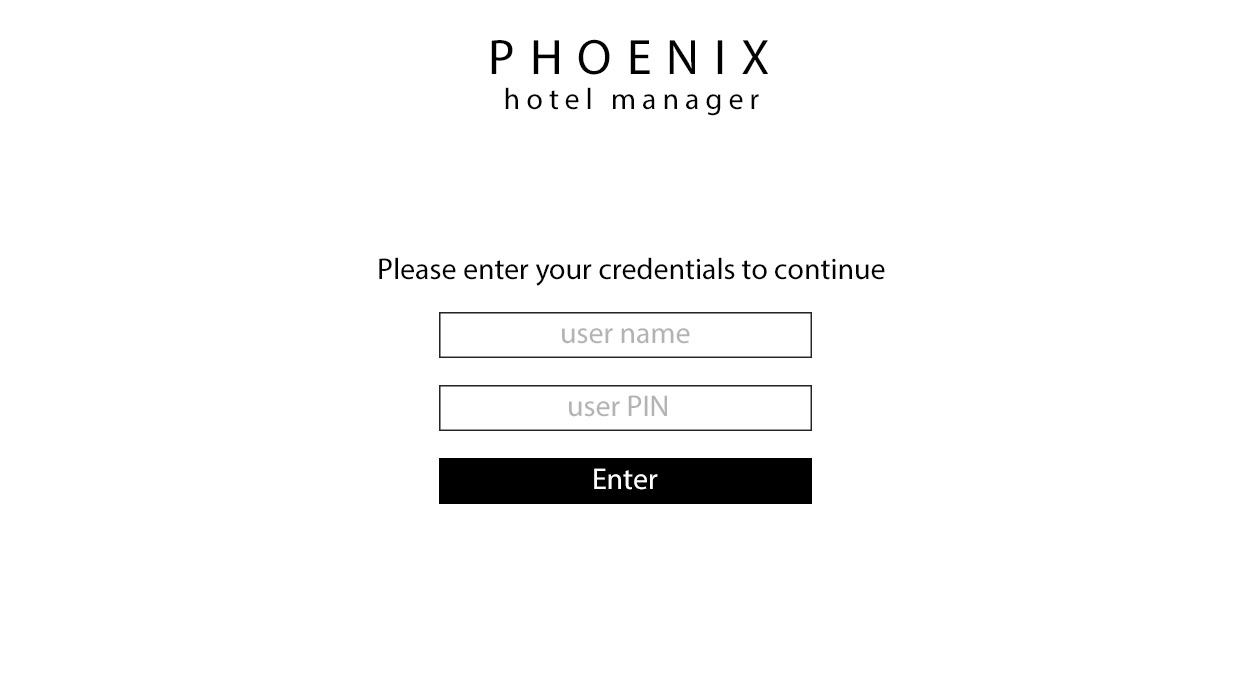
***COCOMO Calculation***

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.

# Prototype



# Project Legacy

Moving forward, there are a variety of ways our project could continue to develop. Since our software was designed in a modular manner, it is a relatively simple process to add functionality without impacting the already existing codebase. In the group presentation, our team discussed the possibility of adding business analytics to keep track of data such as inventory used, total monthly sales, and other statistics created as the software is used. We also discussed the possibility of adding an end-user facing component so that customers would be able to search for available rooms or order room service directly with the software.

As for software maintenance, our code relies on local database that grows as objects such as reservations an invoices are added. It is possible to imagine that after heavy use the database would being impractically large causing performance issues. Maintenance may be required to remove data that is no longer useful and move it to an archive database.

# 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Week:** | **11** |  |  |  |  | **12** |  |  |  |  | **13** |  |  |  |  | **14** |  |  |  |  | **15** |  |  |  |  |
|  | **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** |
| **Presentation** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus: Preparation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew: Preparation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan: Preparation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas: Preparation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj: Slide Deck, Preparation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Final Code** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Final Document** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fergus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Andrew |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ryan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Devidas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Viraj |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# 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.

# Resumes

## Fergus Kelley

[fkelley2@student.gsu.edu](mailto:fkelley2@student.gsu.edu)

Education

* Bachelors of Science, Computer Science, Georgia State University, anticipated graduation Spring 2017

President’s List, 4.0 GPA, Spring 2015 - Spring 2016

* Emergency Medical Technician Certification, Savannah Technical College

Dean’s List, Fall 2012 - Fall 2013

* Associate of Arts, Oxford College of Emory University
* High School Diploma, Lakeside High School, Atlanta, GA

Skills

* Highly experienced in Java, and web design with JavaScript, HTML, and CSS.
* Some experience with Python, C, PHP, and SPARC assembly, as well as computer architecture.
* Familiarity with 3D graphics and user interface design including Three.js and Unreal Engine 4.
* Experienced user of Windows and Linux.
* Highly proficient with the usage and assembly of computer hardware and software.
* Effective written and verbal communication skills.

Projects – github.com/fergk

* *Callay* and *Brillig*: Abstract terrain generation and architecture design in the browser using Three.js and cannon.js.
* *The Approach*: Overhead 3D twin-stick shooter game using Unreal Engine 4.

Professional Experience

**Barista**

Northlake Starbucks, Tucker, Ga, May 2014 - present

Starbucks Kiosk, Kroger, Savannah, GA, March 2011 - April 2012

* Ensured customer satisfaction by providing exceptional, personable service, assisting with product selection, and resolving customer concerns.

**Library Student Worker**

Savannah Technical College Library, Savannah, GA, January 2013 - December 2013

* Assisted patrons with utilizing library collections, internet resources, online databases, and classroom software.
* Served as first point-of-contact for students requiring further institutional information.
* Circulated and recorded library materials used by patrons.
* Maintained library facilities and equipment.

**Student Conservationist**

Woodruff Library, Emory University, Atlanta, GA, May 2008 - December 2009

* Repaired damaged books and materials, including rare and one-of-a-kind volumes.
* Gained competence in the safe use of tools and specialized machinery, such as drill presses, large-scale paper shears, and scalpels.
* Constructed protective enclosures for fragile or at-risk materials.
* Responded to preservation crises in the university's libraries, including mold and water damage to rare materials.

**Library Stacks Student Worker**

Woodruff Library, Emory University, Atlanta, GA, May 2007 - August 2007

* Organized, reshelved, and maintained library materials according to the Library of Congress classification.

## **Andrew An Dang Truong**

**Contact Information**

1852 Central Park Loop Morrow, GA

Atruong8@student.gsu.edu

(678) 764-4265

**Education**

**Morrow High School, Morrow,GA**—High School Diploma

High School Diploma: May 2013

Cumulative GPA: 3.8/4.0

**Georgia State University, Atlanta,GA**—Bachelor of Science, Computer Science

Expected Graduation Date: December 2017

Cumulative GPA: 3.73/4.0

**Work Experience**

**Dotz Company, Atlanta,GA—**Internship

May 2016 – Present

Assembly line worker; took photographs of products to be uploaded to company’s website; design products; project manager

**Skills**

Multitasking

Organization

Language Experience- Vietnamese, Intermediate

Dedication

Determination

Intermediate Java programming

Beginner Unix, C, C++ programming



## Ryan Ocampo

EXPERIENCE

**Georgia State University Library Department,** Atlanta, Georgia — *Student Assistant*

February 2016 – August 2016

Scan documents such as articles, books, and pamphlets for digitization and accessibility on the University Library catalogue; Data entry;

**Georgia State University Student Support Services,** Atlanta, Georgia — *Tutor*

June 2016 - Present

Arrange Tutoring sessions with students who have trouble in particular courses. Tutoring in courses pertaining to fundamental Java programming and research essay writing

EDUCATION

**Georgia State University, Atlanta, GA**— *Bachelor of Science, Computer Science*

Expected Graduation: December 2017

Cumulative GPA: 3.45/4.0

**Luella High School, Locust Grove, Ga** — *HS Diploma*

HS Diploma: May 2015

Cumulative GPA: 3.8/4.0

SKILLS

* **Proficiency in proofreading articles and publications**
* **Intermediate MS Office, Java, ALICE Programming; beginner Assembly, Unix scripting, C**  **Adaptive learning**
* **Multitasking**
* **Organization**
* **Dedication**
* **Detail-oriented**
* **Language experience- French, Intermediate**

Computer Science Course Taken (programming language taught)

* **CSC 2010- Principles of Computer Science (Java)**
* **CSC 2310- Principles of Computer Programming(Java)**
* **CSC 3210- Computer Organization and Programming(Assembly Code)**
* **CSC 3410- Data Structures(Java)**
* **CSC 3320- System Level Programming(Unix scripting, C)**

## Deividas Rutkauskas

10 Perimeter Park Dr • Atlanta, Georgia • 30341

**CELL** 404 731 1443• **E-MAIL** deividasrr@yahoo.com

!

EDUCATION **Georgia State University Expected graduation: 2017**

Atlanta, GA

Bachelor of Science, Computer Science

Current GPA: 3.79

**East Coweta High School 2013**

Sharpsburg, GA

Zell Miller Scholarship Recipient

!

SKILLS **Web development**

* Adept at UI and visual design
* Good working knowledge of HTML and CSS, and Javascript
* Experience with WebGL and Babylon.js
* Introductory knowledge of PHP, MySQL and Apache

**Computer programming**

* **I**ntermediate Java skills
* Some familiarity with Python

**Art and Graphic Design**

* Skilled in drawing
* Digital post-processing skills

**Proficiency in Microsoft Office, Apple iWork**

* Completed numerous projects with the software

**Photoshop, CorelDRAW**

* Created advertisements, post-processed photoshoots
* Modeled website mockups in PS

**Semi-professional Photography**

* Shot numerous friends and acquaintances to build up experience
* Strong knowledge of both DSLR and Film photography
* Photo portfolio can be found [**here**](https://www.flickr.com/photos/undamurmur/)

**Communication**

* Fluent in English and Lithuanian
* Conversational knowledge of French
* Good public speaking skills

!

EXPERIENCE **Seasonal Sales Associate at Abercrombie & Fitch**

**10/2014-01/2015**

* Performed such tasks as keeping the store tidy, tagging and replenishing items, servicing customers and answering their questions
* Gained experience in working in an extremely busy environment with a substantial volume of customers

**Volunteer tutor at a primary school via National Honor Society**

**02/2012-06/2012**

* Helped students with vocabulary and grammar
* Cultivated a patient, friendly and helpful demeanor

**Freelance web designer**

**02/2012-06/2012**

* Coordinated work with a web programmer
* Developed unique designs according to the customer’s requests
* Created posters, flyers and business cards
* Significantly increased knowledge of Photoshop and vector-based software

**10 Perimeter Park Dr** • Atlanta, Georgia • 30341• **CELL** 404 731 1443 • **E-MAIL** deividasrr@yahoo.com

## **Viraj Shah**

3226 Bagley Passage, Duluth, Georgia 30097 • 770-401-0751 • Hello@VirajAShah.com • VirajAShah.com

PROFESSIONAL PROFILE

*I am an aspiring software professional with an affinity for learning and applying new technology stacks to garner tangible results. Eager to pursue employment with ambitious companies to improve my industry knowledge and develop worthwhile software solutions.*

TECHNICAL SKILLS

**SOFTWARE:** C#, WPF, JAVA, OBJECTIVE-C, SWIFT( Native iOS development), HTML5,

CSS3, JAVASCRIPT ( Bootstrap Framework), JSON, Visual Studio 2012, XCode, IntelliJ

**DATABASE:** SQLite, mySQL

**ARCHITECTURES:** Windows Azure/Cloud Services, OOA, OOD

**HARDWARE & OS:** Windows 7/8/10, iOS 9, VMWare 5.2, Workstation 9, Unix, Linux

EDUCATION

GEORGIA STATE UNIVERSITY Atlanta, Georgia

**Bachelor of Science in Computer Science** *Expected Graduation December 2016*Computer Software Systems Concentration and a Minor in Mathematics

WORK EXPERIENCE

TRUSTPOINT SOLUTIONS Atlanta, Georgia

***IT Security Consultant*** *May 2016 – Present*

*Conducted Risk Assessments and Risk Analysis’ on all applications containing ePHI and sensitive information in compliance with HIPAA. Performed assessments on client software development processed*

*.*

NAVLIT Atlanta, Georgia iOS Developer April 2016 – August 2016

*Lead iOS Developer for startup company Navlit responsible for deploying a native iOS application in Swift and integrating backend with Azure Web Services. Responsible for creating and integrating web service APIs into mobile application and maintaining and updating the mobile app.*

FREELANCE WEB DEVELOPER Atlanta, Georgia

***Front End Developer*** *January 2016 – Present*

*Contract based work developing websites for small businesses with a focus on being mobile friendly. Websites created using the Bootstrap Framework ( HTML,CSS, JS ) and hosted with a LAMP STACK.*

*VeronicaShah.com VirajAShah.com VetekConsulting.com AmericasDreamEstates.com NavLit.com*

ITNEER Buford, Georgia Software Development Intern May 2015 – August 2015

*Primarily worked as a junior software engineer helping develop iOS applications using Objective-C and Swift. Gained familiarity developing mobile solutions and exposure to mobile frameworks.*

TOUCH THE EARTH Atlanta, Georgia

**Staff *Member*** *May 2014 – December 2015*

*Provided expertise and knowledge to patrons regarding outdoor excursions. Qualified trip leader with experience leading multiple university sanctioned expeditions. Trained in First Aid, CPR, and Belay certification.*

INNOVATIVE ARCHITECTS Duluth, Georgia

**Software *Development Intern*** *May 2011 – August 2011*

*Worked as an intern responsible for extensive testing and debugging of deployed software products as well as documenting new software features.*

# User Guide

## Installation and setup

PRMS can be started by running the .jar file. If the software has not been run before, a blank database will be created. This database will be empty except for an admin employee profile.

In order to log-in for the first time, use the following credentials:

Username: admin

Password: admin

For system security, after logging in for the first time, you should select the Employees tab and create a new account with your information. You can then delete the admin account to prevent others from logging in with that account.

## Employees

From the Employees tab, you can create, modify, or remove employee profiles. The left half of this tab contains the controls for managing the employee profiles. The right side of the tab is a table displaying all the employee profiles currently in the database. The columns of the table can be sorted by clicking on the header, and rearranged by dragging.

To create an employee profile, fill in each of the fields on the left of the screen. The value in username field must be unique and cannot be the same as the username of any existing employee profiles. All fields are required, and the two password fields must match. After filling each of the fields, hit apply to create the profile. It will appear in the table to the right.

To modify or remove an existing employee profile, click the modify or remove radio button and type the username into the username field or click on the profile in the table to the right. All fields but the username can be modified. Hit apply to modify or remove the profile.

Hotel Rooms

From the Hotel Rooms tabs, you can create, modify, or remove the hotel rooms in the resort. Similar to the Employees tab, you can enter information into the fields on the left and view the table of rooms to the right.

The floor and number fields are combined to create the full room number that each hotel room is identified by. The format is FFNN where FF is the floor number of the room, and NN is the number of that room on the floor. As an example, the 4th room on the 4th floor would be room number 404. The floor and number fields can take values between 1 and 99 inclusive.

The floor and number fields both accept ranges for managing more than one room at a time. For example, to create a hotel with 5 floors and 10 rooms on each floor, you would enter “1-5” into the floor field and “1-10” into the number field. This would create rooms numbered 101, 102, 103 … 508, 509, 510.

The price field contains the amount the room costs per night, in dollars and cents. Any value with more than 2 decimal places to the right of the decimal point will be rounded to the nearest cent. The bed field takes an integer representing the number of beds in the room. The three check boxes indicate whether the room allows pets, allows smoking, or is disability accessible.

As with the Employees tab, you can modify or remove hotel rooms by selecting the modify or remove buttons and entering in floor or number fields, or you can select a room from the list to the right.

## Event Rooms

This tab behaves almost exactly like the Hotel Rooms tab, except that it manages event rooms that a resort might offer for conferences or gatherings.

Unlike hotel rooms, event rooms are identified by unique names and contain fields for whether the room contains a stage and audio/visual equipment.

## Hotel Maintenance

From this tab, you can manage the cleaning and inventory of hotel rooms. On the left is a list of all the rooms and the date they were last cleaned. Selecting a room and pressing the “update date last cleaned” button will update the status of the hotel room. This can be used by staff to keep track of which rooms need cleaning.

By selecting a room in the left table, the right table will list the inventory contained in that room. This might include, towels, televisions, etc. Items can be selected and their quantity updated by clicking on the item in the table and putting a value in the text field below.

## Hotel Reservations

This tab is used for reserving rooms in the hotel. Check-in and check-out dates can be selected, along with a variety of room properties. The table on the right contains a list of rooms that fulfill that search query.

## Event Bookings

This tab works similar to the Hotel Reservations tab, however it is used for creating bookings for the event rooms.

## Invoices

This tab handles editing and modification of invoices that are created with reservations, bookings, and orders. The left lists the currently created invoices. The right table contains the list of all the items contained on the selected invoice.

## Orders

## Restaurant