

CRYSTAL PALACE

HOTEL MANAGEMENT SYSTEM



Spring 2017
Georgia State University

Final Deliverable

4/24/2017

Team Members:

James Gangavarapu

Joshua Katikala

Robert Kim

Eldin Mulalic

Eden Taitt

Jason Tatum

Table of Contents

1. Purpose of this Application	2
2. Actors	2
3. Functional Requirements	2
3.1 Login	2
3.2 Reservations	2
3.3 Employee Access	2
3.4 Manager Access	3
4. Non-Functional Requirements	3
4.1 Usability	3
4.2 Reliability	3
4.3 Performance	4
4.4 Supportability	4
4.5 Implementation	4
4.6 Interface Requirements	4
4.7 Packaging Requirements	4
4.8 Legal Requirements	4
4.9 Security	4
4.10 Scalability	4
5. Requirements Traceability Matrix	5
6. Database	10
7. System Model	11
7.1 Use Case Model	11
7.2 Object Model	18
7.3 Dynamic Model	20
8. System Rationale	28
8.1 System Architecture Rationale	28
8.2 Object Design Rationale	28
9. Test Cases	29
10. Horizontal Prototype	38
11. Cost Analysis	40
11.1 Function Point Cost Analysis	40
11.2 Construction Cost Model (COCOMO)	41
11.3 FPA and COCOMO Analysis	42
12. Work Structure Document	43
13. Gantt Chart	44
14. Dictionary of Terms	45
15. Current Code	46
16. Legacy	47
17. User Guide	48
18. Team Member Resumes	57

1. Purpose of this Application

This application solves the problem of a manual hotel management system having to manually gauge and keep track of room reservations, and supplies thereof. It also provides a tool for managers to view meaningful data about employees that would aid in managing them, mostly through viewing of the most relevant data needed to reach a management decision or evaluate performance.

2. Actors

There are three actors in this system: the employees (receptionist and supervisor), the manager, and the database.

3. Functional Requirements

3.1 Login

3.1.1 The system shall have three main access levels: receptionist, supervisor, and manager.

3.1.2 All employees shall be able to log into the system using their unique credentials.

3.1.3 The system shall keep track of the hours worked by an employee from login to logout time.

3.2 Reservations

3.2.1 The system shall have modifiable types of rooms attributed to rooms at a hotel.

3.2.2 The system shall include an optional component to display the number of check-ins and check-outs remaining for the day as well as the current occupancy rate.

3.2.3 The system shall have an option to enable alerts to particular employees if guests are late or missed check-ins.

3.2.4 The system shall include two types of accounts for room reservations: cash and credit. The system will not allow room service charges to be added to a cash account. Additionally, cash accounts will require a deposit which will be deducted from the final balance as applicable during checkout.

3.3 Employee Access

3.3.1 The employee shall be able to make reservations for guests.

3.3.2 The employee shall be able to check for room availability within a time period.

3.3.3 The employee shall be able to provide cost estimates before confirming a reservation.

3.3.4 The employee shall be able to check in and check out guests.

3.3.5 The employee shall be able to cancel reservations.

3.3.6 The employee shall be able to input room service orders.

- 3.3.7** The employee shall be able to access guest costs and balances.
- 3.3.8** The employee shall be able to generate invoices and process payments.
- 3.3.9** The employee shall be able to access the guest table in the database which keeps track of guests' personal information, payment information, loyalty points, and any other relevant information.
- 3.3.10** The employee shall be able to add new guests to the database.
- 3.3.11** The employee shall be able to access returning guests' information.
- 3.3.12** The employee shall be able to extend guest stays based on availability.
- 3.3.13** The employee shall be notified about guest-requested wake-up calls.

3.4 Manager Access

- 3.4.1** The manager shall be able to log into the manager access using his/her unique credentials.
- 3.4.2** The manager access level shall include the employee information, employee time sheets, payroll information, inventory records, and finances.
- 3.4.3** The manager shall be able to add or remove employees.
- 3.4.4** The manager shall be able to manage employee work schedules.
- 3.4.5** The manager shall be able to adjust employee base pay.
- 3.4.6** The manager shall be able to view system-generated weekly and monthly business reports based on occupancy rates.
- 3.4.7** The manager shall be able to alter room costs.
- 3.4.8** The manager shall be able to override guest balances.
- 3.4.9** The manager shall receive notifications periodically suggesting hotel maintenance requirements such as exterminator and landscaping appointments.
- 3.4.10** The manager shall have access to an inventory database which will keep track of bedding, towels, toiletry items, cleaning supplies, room service items, and all other relevant inventory.
- 3.4.11** The manager shall receive warnings for low stock items.
- 3.4.12** The manager shall have access to a phonebook database which will keep track of important numbers such as suppliers and maintenance companies.
- 3.4.13** At the end of each day, the manager shall be able to view a list of all employee clock-in and clock-out times for payroll purposes.
- 3.4.14** The manager shall be able to generate coupon codes.

4. *Non Functional Requirements*

4.1 Usability

- 4.1.1** CPMS shall provide an easy to use interface that requires minimal training for use.
- 4.1.2** CPMS shall provide a user manual for additional help if necessary.
- 4.1.3** CPMS shall be accessed using a mouse and keyboard combination.

4.2 Reliability

- 4.2.1** CPMS shall be capable of operating at all times.

4.2.2 CPMS shall handle system exceptions via popup windows.

4.2.3 CPMS shall restrict access to the system only to employees and managers.

4.3 Performance

4.3.1 CPMS shall provide a quick and responsive interface.

4.3.2 CPMS shall generate business, tax, or other broad data reports in a timely manner.

4.4 Supportability

4.4.1 CPMS shall be platform independent since it is written in Java and SQL.

4.4.2 CPMS shall not require IT support for maintenance.

4.5 Implementation

4.5.1 CPMS shall require an enterprise level computer to operate on.

4.6 Interface Requirements

4.6.1 CPMS shall not require interaction with any existing systems. The only actors are the users and the database.

4.6.2 CPMS shall utilize the user interface to import and export all data in the system.

4.7 Packaging requirements

4.7.1 CPMS shall be installed by a Crystal Palace Team Member and is meant for hotel internal use only.

4.8 Legal Requirements

4.8.1 CPMS shall be installed on only one system per license.

4.8.2 CPMS shall not be liable for any system failures.

4.8.3 CPMS shall not be repurposed or resold for any reason.

4.8.4 CPMS shall not be responsible for any data theft due to system insufficiencies.

4.9 Security

4.9.1 CPMS shall partition access between employees and managers.

4.10 Scalability

4.10.1 CPMS shall be capable of running on multiple computers independently.

5. Requirements Traceability Matrix

Entry #	Section #	System Specification	Type	Use Case Name
3.1.1	3.1	The system shall have three main access levels: receptionist, supervisor, and manager.	SW	N/A
3.1.2	3.1	All employees shall be able to log into the system using their unique credentials.	SW	Login Use Case
3.1.3	3.1	The system shall keep track of the hours worked by an employee from login to logout time.	SW	Clock In Use Case Clock Out Use Case
3.2.1	3.2	The system shall have modifiable types of rooms attributed to the rooms at a hotel.	SW	N/A
3.2.2	3.2	The system shall include an optional component to display the number of check-ins/checkouts remaining for the day as well as the current occupancy rate.	SW	View Reservations Use Case
3.2.3	3.2	The system shall have an option to enable alerts to particular employees if guests are late or missed check-ins.	SW	N/A
3.2.4	3.2	The system shall include two types of accounts for room reservations: cash and credit. The system will not allow room service charges to be added to a cash account. Additionally, cash accounts will require a deposit which will be deducted from the final balance as applicable during checkout.	SW	New Reservations Use Case
3.3.1	3.3	The employee shall be able to make reservations for guests.	SW	New Reservations Use Case
3.3.2	3.3	The employee shall be able to check for room availability within a time period.	SW	N/A
3.3.3	3.3	The employee shall be able to provide cost estimates before confirming a reservation.	SW	New Reservations Use Case
3.3.4	3.3	The employee shall be able to check in and check out guests.	SW	Guest Search Use Case
3.3.5	3.3	The employee shall be able to cancel	SW	Guest Search Use

		reservations.		Case
3.3.6	3.3	The employee shall be able to input room service orders into the system.	SW	Guest Search Use Case
3.3.7	3.3	The employee shall be able to access guest costs and balances.	SW	Guest Search Use Case
3.3.8	3.3	The employee shall be able to generate invoices and process payments.	SW	N/A
3.3.9	3.3	The employee shall be able to access guest tables in the database which keep track of guests' personal information, payment information, loyalty points, and any other relevant information.	SW	Guest Search Use Case
3.3.10	3.3	The employee shall be able to add new guests to the database.	SW	New Reservation Use Case
3.3.11	3.3	The employee shall be able to access returning guests' information.	SW	Guest Search Use Case
3.3.12	3.3	The employee shall be able to extend guest stays based on availability.	SW	New Reservation Use Case
3.3.13	3.3	The employee shall be notified about guest-requested wake-up calls.	SW	N/A
3.4.1	3.4	The manager shall be able to log into the manager access using his/her unique credentials.	SW	Login Use Case
3.4.2	3.4	The manager access level shall include employee information, employee time sheets, payroll information, inventory records, and finances.	SW	N/A
3.4.3	3.4	The manager shall be able to add or remove employees.	SW	View Employees Use Case
3.4.4	3.4	The manager shall be able to manage employee work schedules.	SW	N/A
3.4.5	3.4	The manager shall be able to adjust employee base pay.	SW	View Employee Use Case

3.4.6	3.4	The manager shall be able to view system-generated weekly and monthly business reports based on occupancy rates.	SW	N/A
3.4.7	3.4	The manager shall be able to alter room costs.	SW	Guest Search Use Case
3.4.8	3.4	The manager shall be able to override guest balances.	SW	Guest Search Use Case
3.4.9	3.4	The manager shall receive notifications periodically suggesting hotel maintenance requirements such as exterminator and landscaping appointments.	SW	N/A
3.4.10	3.4	The manager shall have access to the inventory tables in the database which will keep track of bedding, towels, toiletry items, cleaning supplies, room service items, and all other relevant inventory.	SW	-Inventory Use Case -Add Inventory Use Case
3.4.11	3.4	The manager shall receive warnings for low stock items.	SW	N/A
3.4.12	3.4	The manager shall have access to the phonebook table in the database which will keep track of important numbers such as suppliers and maintenance companies.	SW	N/A
3.4.13	3.4	At the end of each day, the manager shall be able to view a list of all employee clock-in and clock-out times for payroll purposes.	SW	N/A
3.4.14	3.4	The manager shall be able to generate coupon codes.	SW	N/A
4.1.1	4.1	CPMS shall provide an easy to use interface that requires minimal training for use.	SW	N/A
4.1.2	4.1	CPMS shall provide a user manual for additional help if necessary.		N/A
4.1.2	4.1	CPMS shall be accessed using a mouse and keyboard combination.		N/A

4.2.1	4.2	CPMS shall be able to operate at all times		N/A
4.2.2	4.2	CPMS shall handle system exceptions via popup windows.	SW	N/A
4.2.3	4.2	CPMS shall restrict access to the system only to employees and managers.	SW	N/A
4.3.1	4.3	CPMS shall provide a quick and responsive interface.	SW	N/A
4.3.2	4.3	CPMS shall generate business, tax, or other broad data reports in a timely manner.	SW	N/A
4.4.1	4.4	CPMS shall be platform independent since it is written in Java and SQL.	SW	N/A
4.4.2	4.4	CPMS shall not require IT support for maintenance.		N/A
4.5.1	4.5	CPMS shall require an enterprise level computer to operate on.		N/A
4.6.1	4.6	CPMS shall not require interaction with any existing systems. The only actors are the users and the database.		N/A
4.6.2	4.6	CPMS shall utilize the user interface to import and export all data in the system.	SW	N/A
4.7.1	4.7	CPMS shall be installed by a Crystal Palace Team Member and is meant for hotel internal use only.		N/A
4.8.1	4.8	CPMS shall be installed on only one system per license.		N/A
4.8.2	4.8	CPMS shall not be liable for any system failures.		N/A
4.8.3	4.8	CPMS shall not be repurposed or resold for any reason.		N/A
4.8.4	4.8	CPMS shall not be responsible for any data		N/A

		theft due to system insufficiencies.		
4.9.1	4.9	CPMS shall partition access between employees and managers to ensure security.	SW	N/A
4.10.1	4.10	CPMS shall be capable of running on multiple computers independently.	SW	N/A

6. *Database*

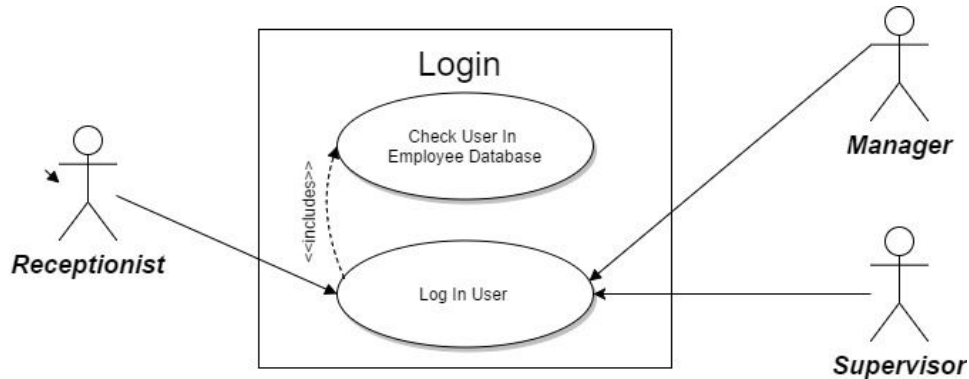
The database to be used is SQLite, chosen based on the constraints of having to use Java in this course.

7. System Model

7.1 Use Case Model

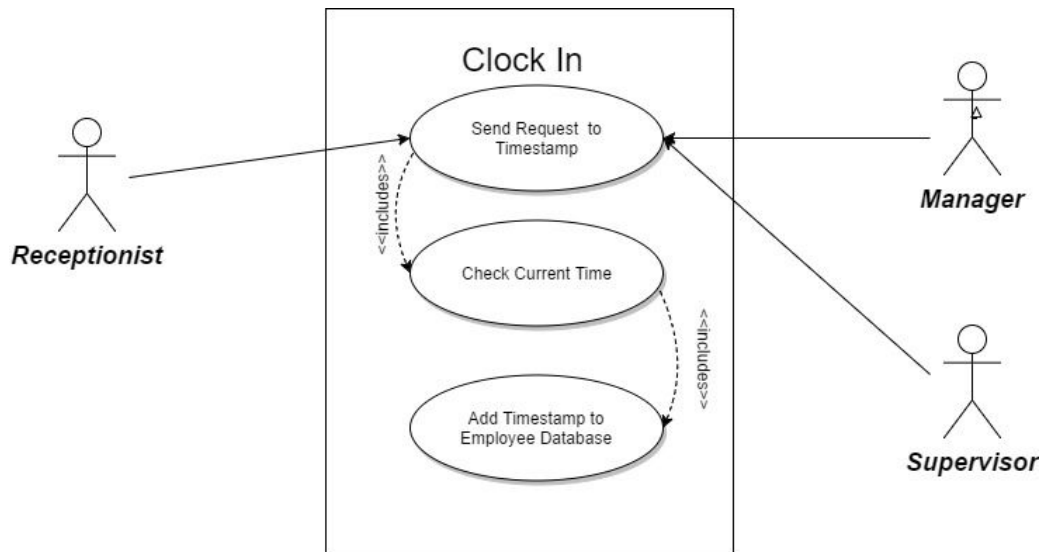
7.1.1 Login Use Case Diagram

This lets all users log in. Managers can check other employees in the database as well as clock other employees out.



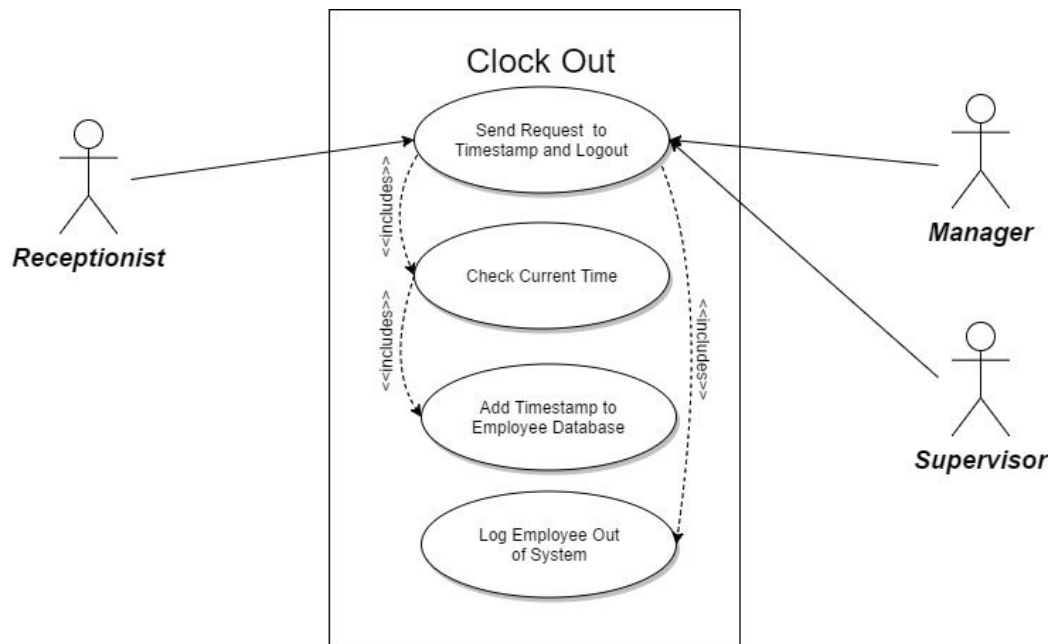
7.1.2 Clock In Use Case Diagram

This lets users clock in, and the timestamp is saved in the database.



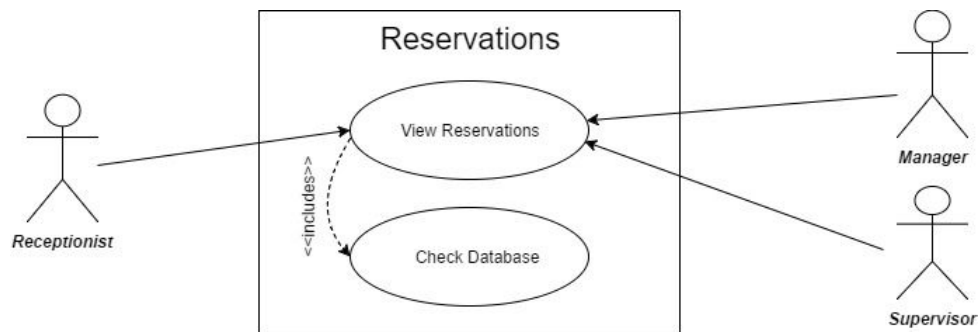
7.1.3 Clock Out Use Case Diagram

This lets users log out of the system, and the timestamp is saved in the database.



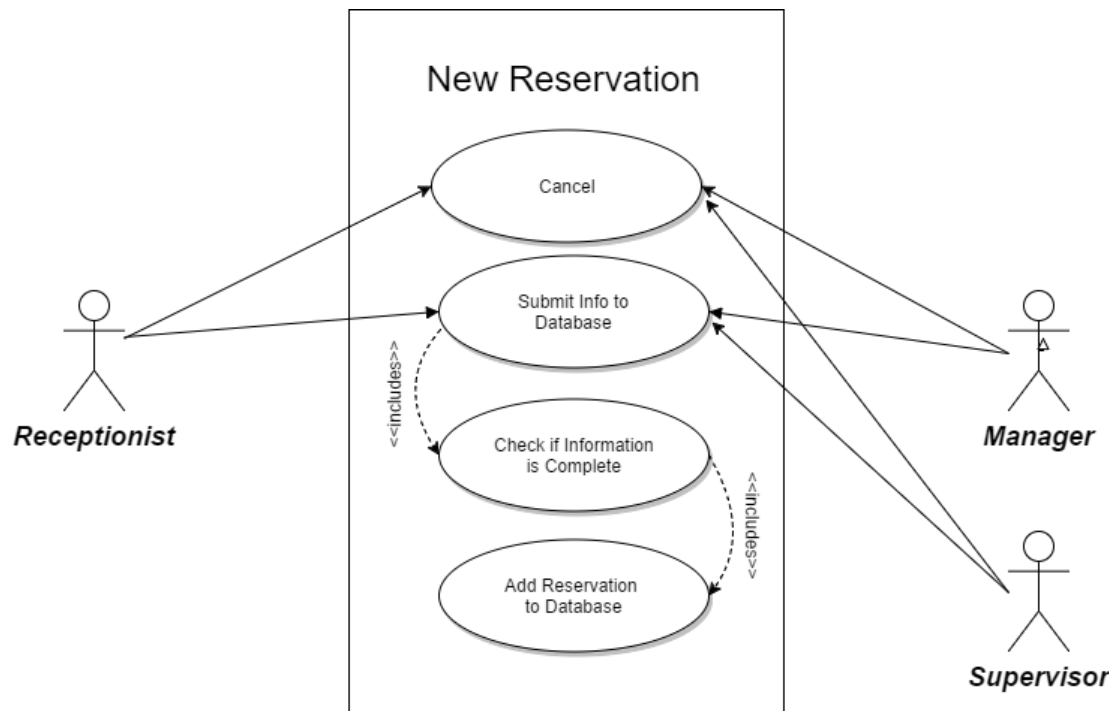
7.1.4 Reservation Use Case Diagram

This lets users view reservations.



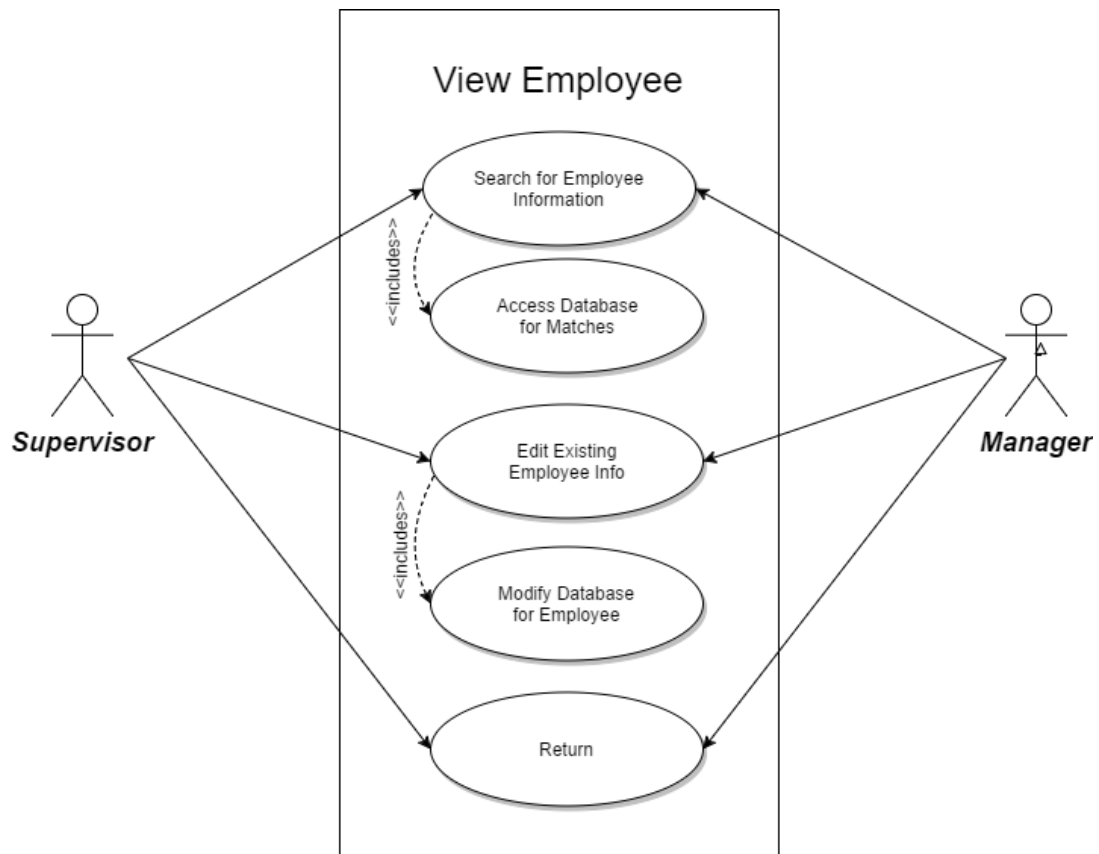
7.1.5 New Reservation Use Case Diagram

This lets users add a new reservation by filling out the necessary information and clicking "Submit."



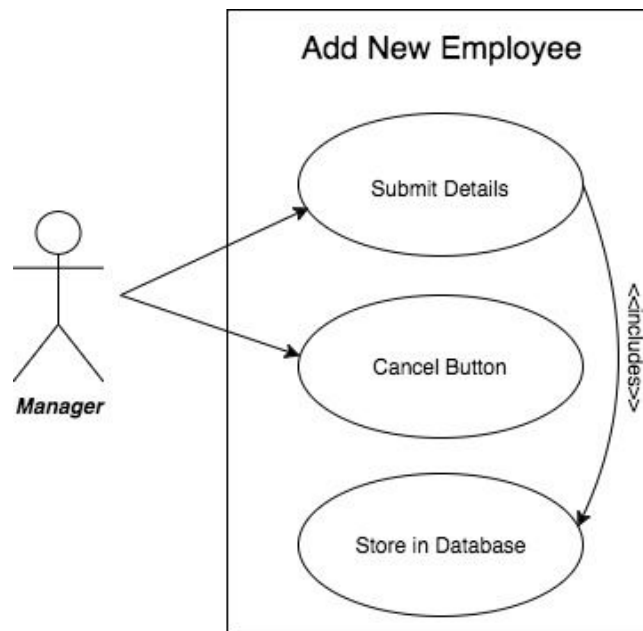
7.1.6 View Employees Use Case Diagram

This is restricted only to the supervisor and manager access levels. It lets the user search for employee information and edit existing employee information



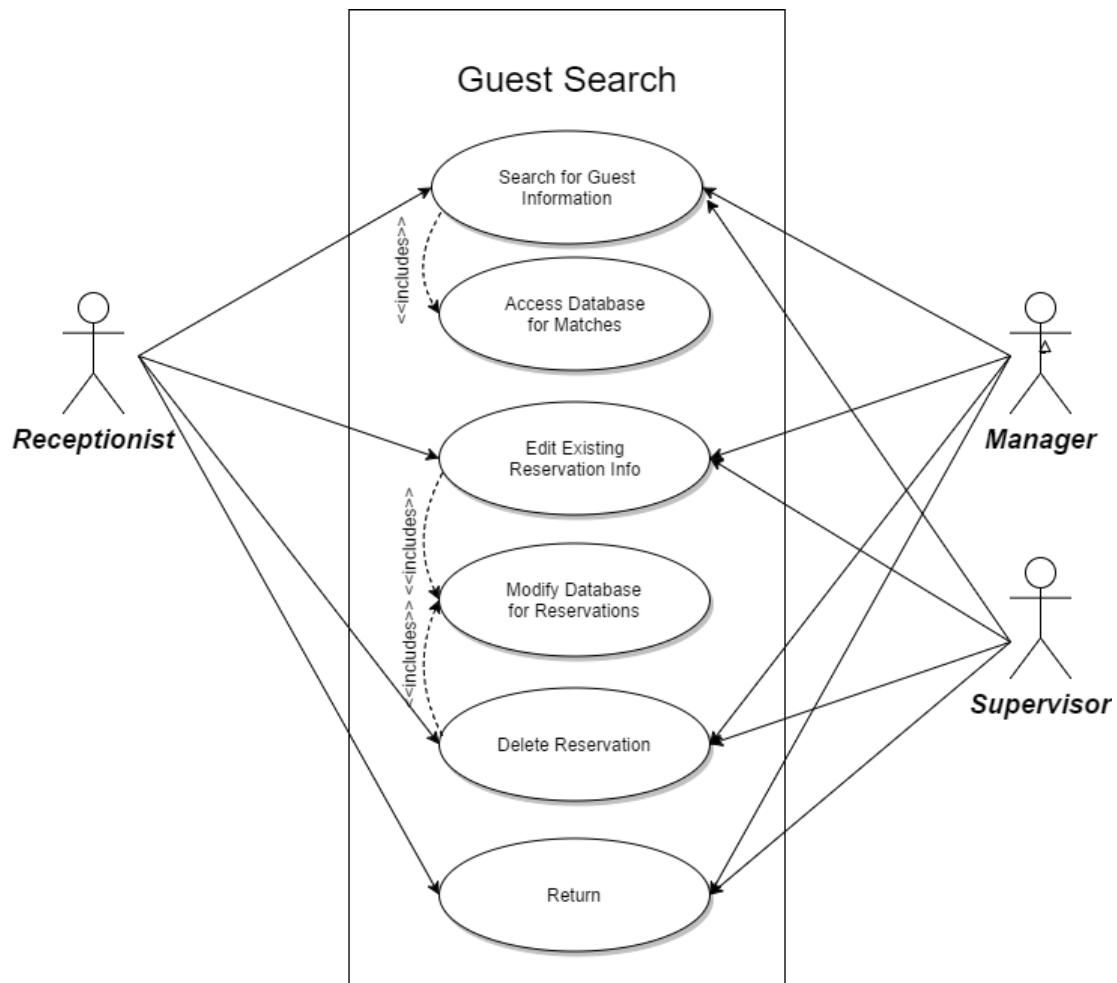
7.1.7 Add New Employee Use Case Diagram

This is restricted to the manager access level only, and it lets the user add an employee by filling out the necessary information and clicking “Submit.”



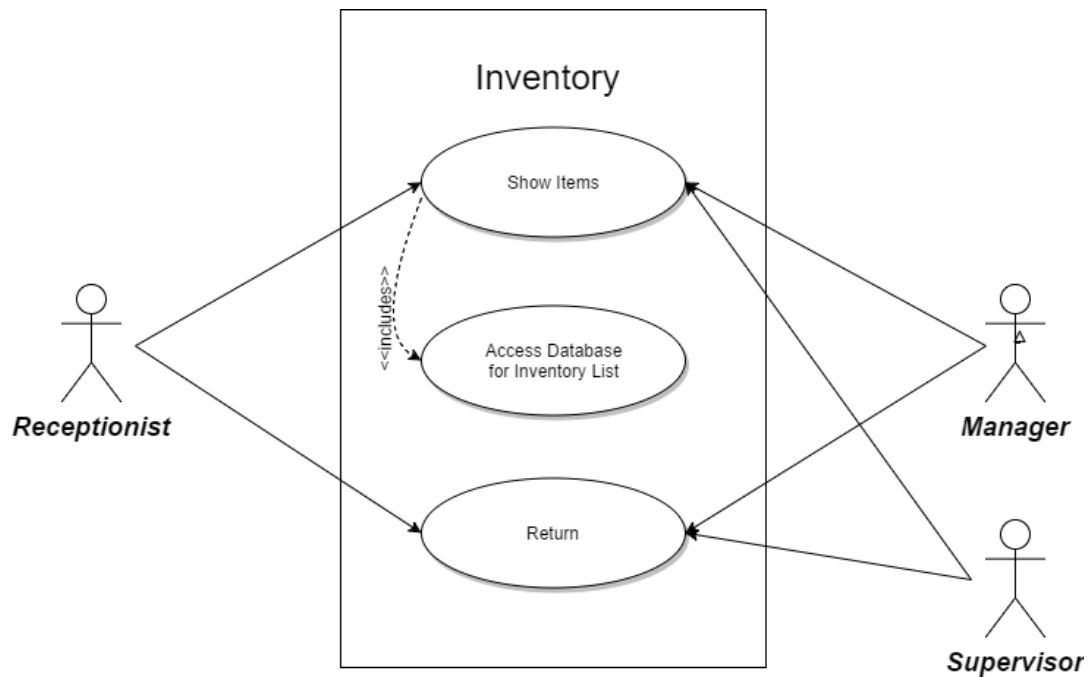
7.1.8 Guest Search Use Case Diagram

This lets all users search for a guest, edit reservation information (including room service), and delete a reservation.



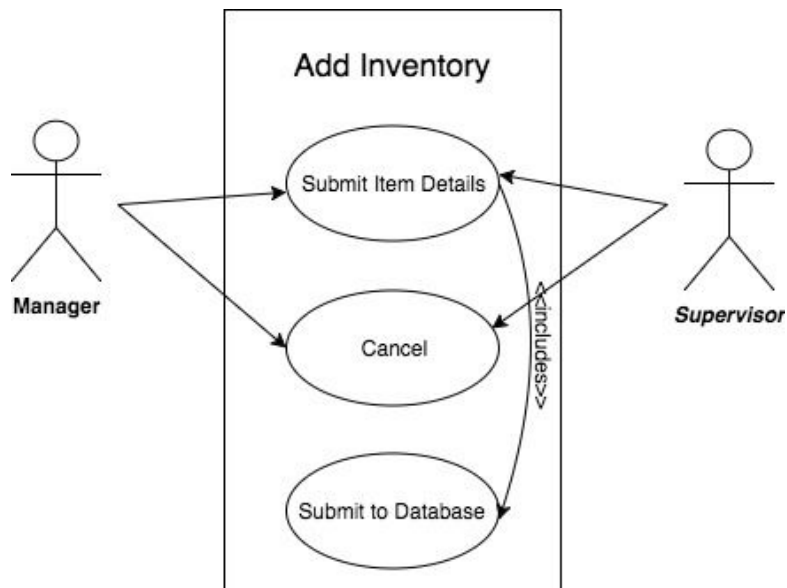
7.1.9 Inventory Use Case Diagram

This lets users view inventory.



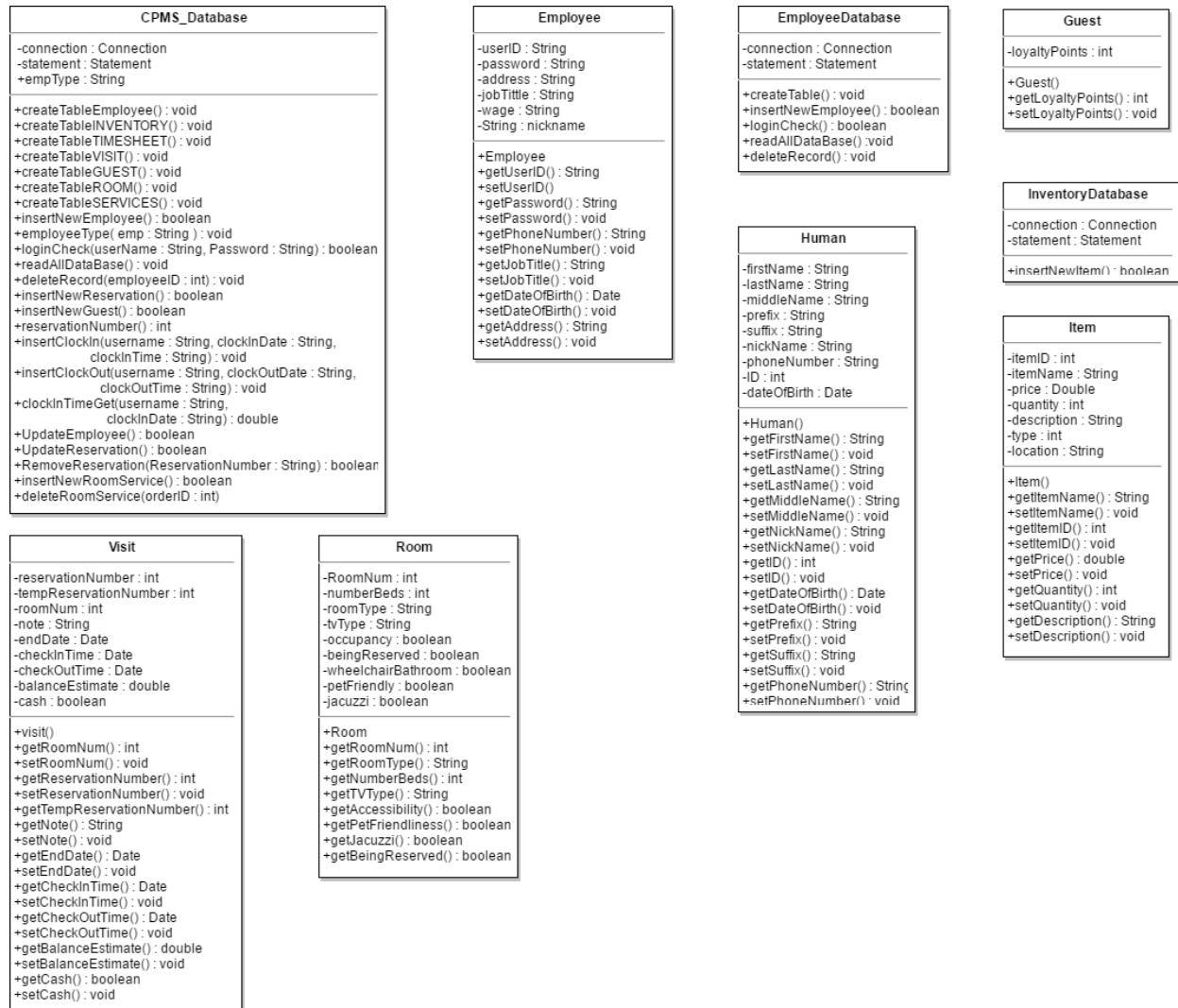
7.1.10 Add Inventory Use Case Diagram

This is restricted only to the supervisor and manager access levels, and it lets the user add inventory by filling out necessary information and clicking "Submit."



7.2. Object Model





7.2.1 Object Model Rationale

The two diagrams for the object models were separated for visibility. The only way to neatly catalog the interaction would be to have them all on one such diagram but since they are separated for visibility there are none. The key interaction is that of the *controller* classes to the *CPMS_Database* and then back to the FXML Views.

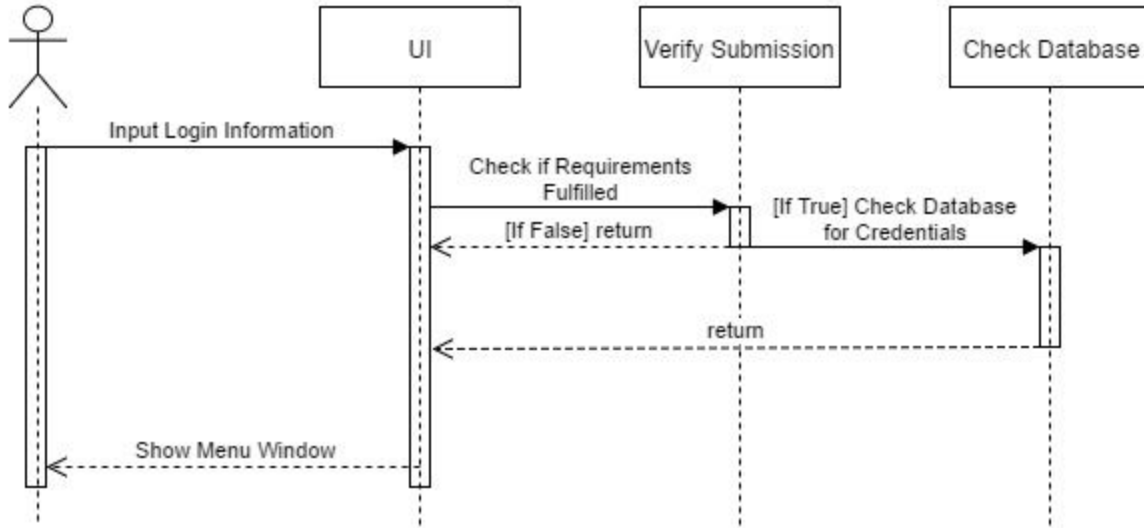
7.3 Dynamic Model

7.3.1 Use Case Sequence Diagrams

The following Sequence Diagrams aid in explaining the behavior of various operations performed by the system and by its actors.

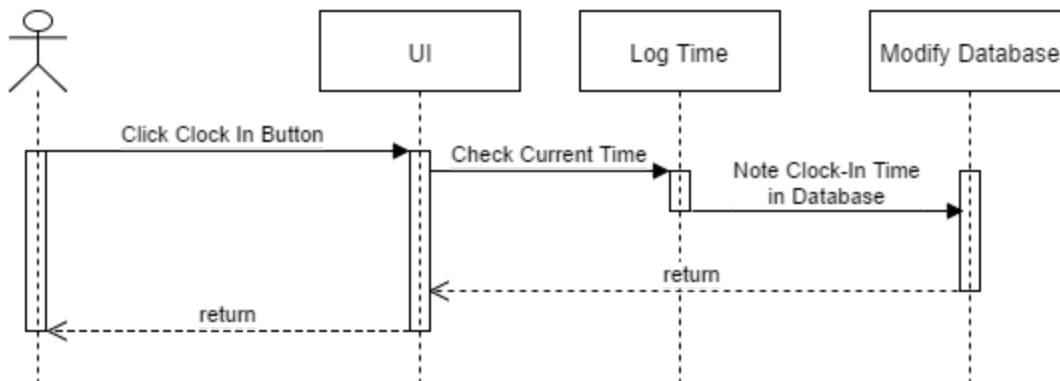
7.3.1.1 Login Sequence

Login Sequence



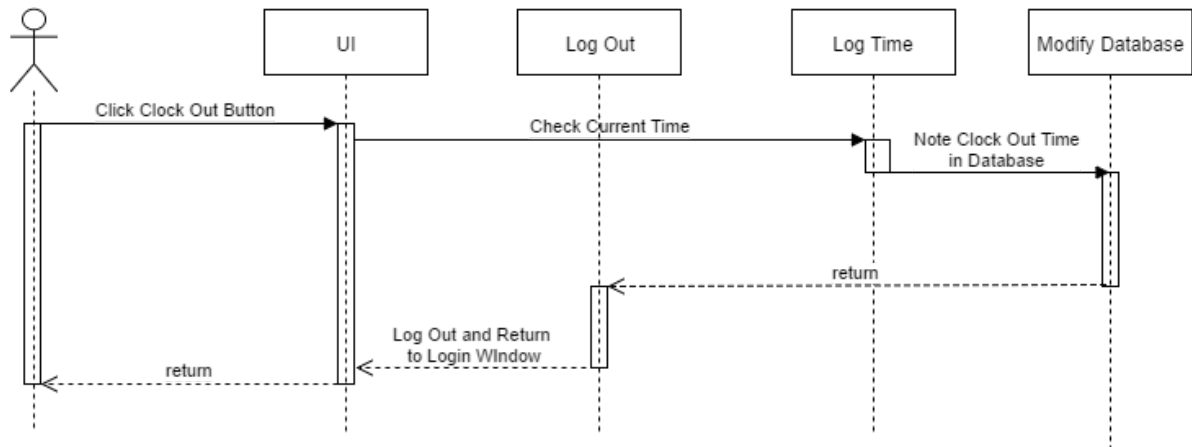
7.3.1.2 Clock In Sequence

Clock In Sequence



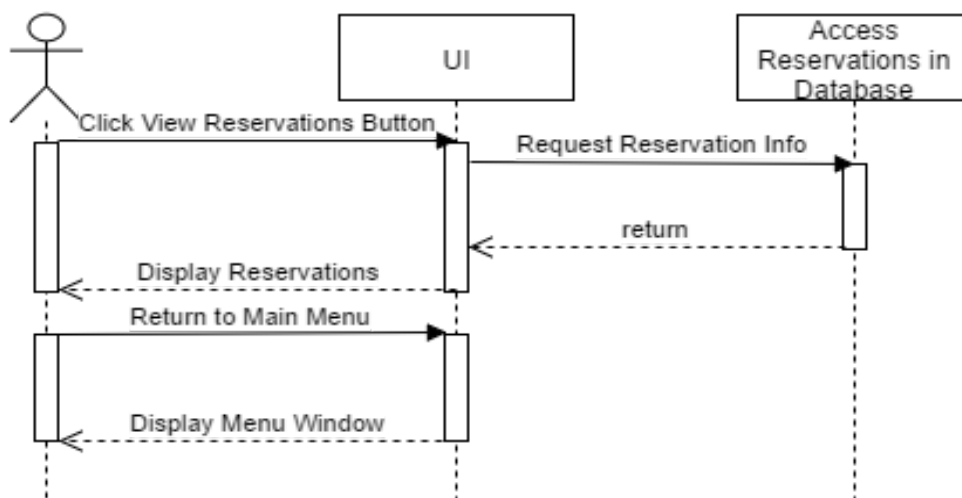
7.3.1.3 Clock Out Sequence

Clock Out Sequence



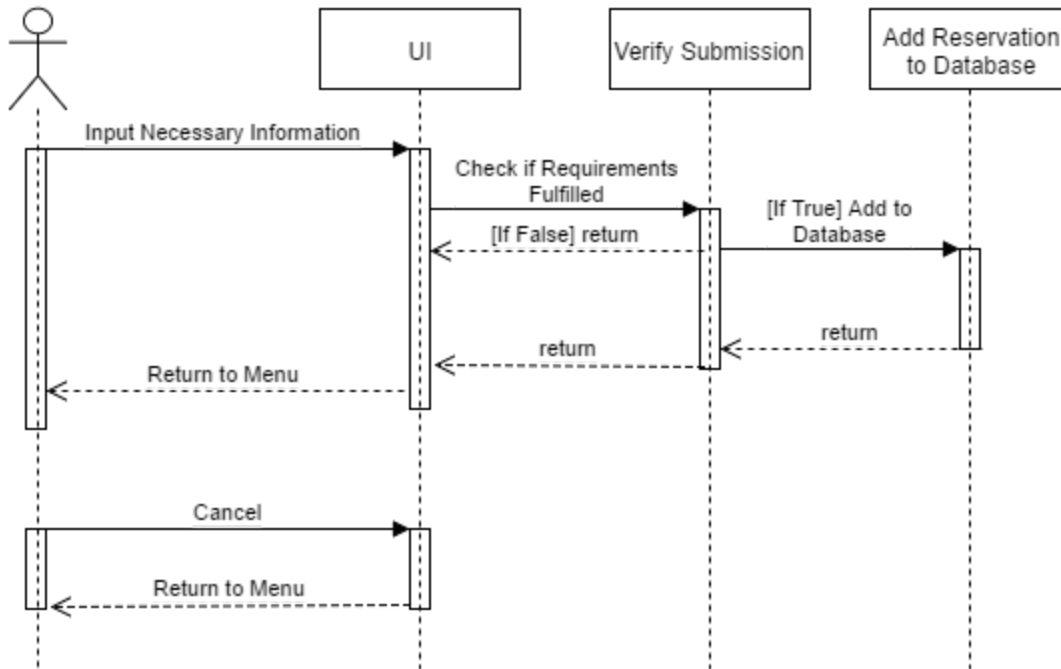
7.3.1.4 Reservations Sequence

Reservations Sequence



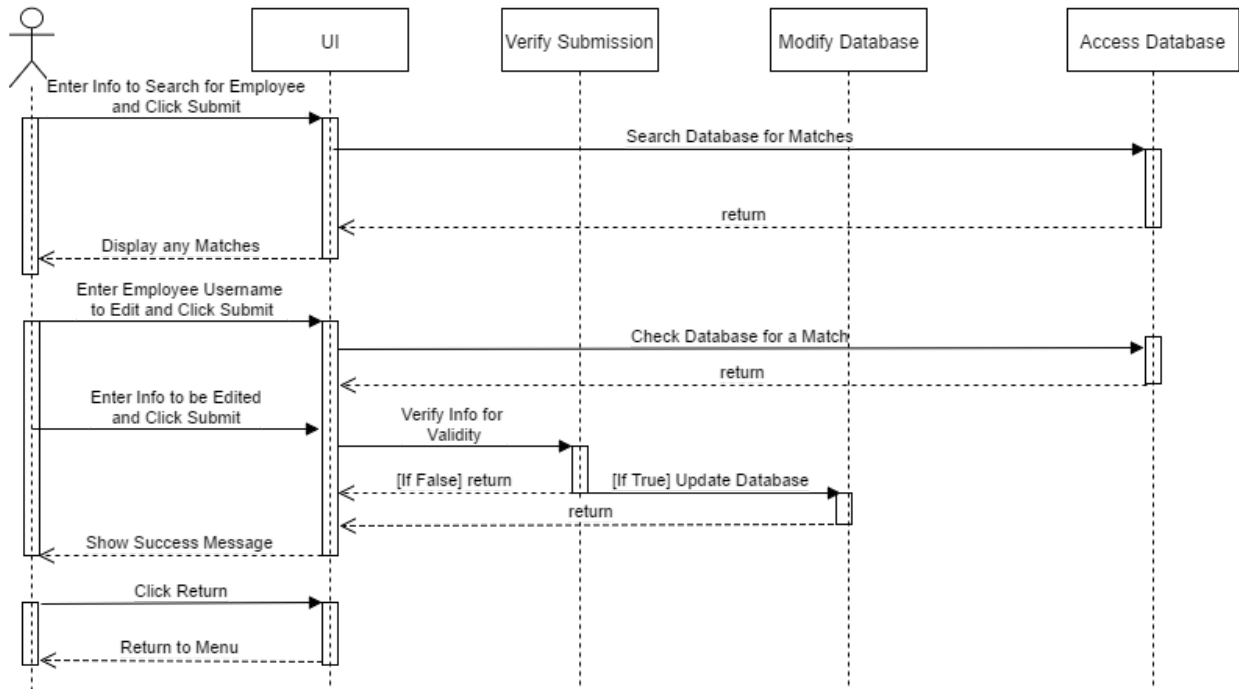
7.3.1.5 New Reservation Sequence

New Reservation Sequence



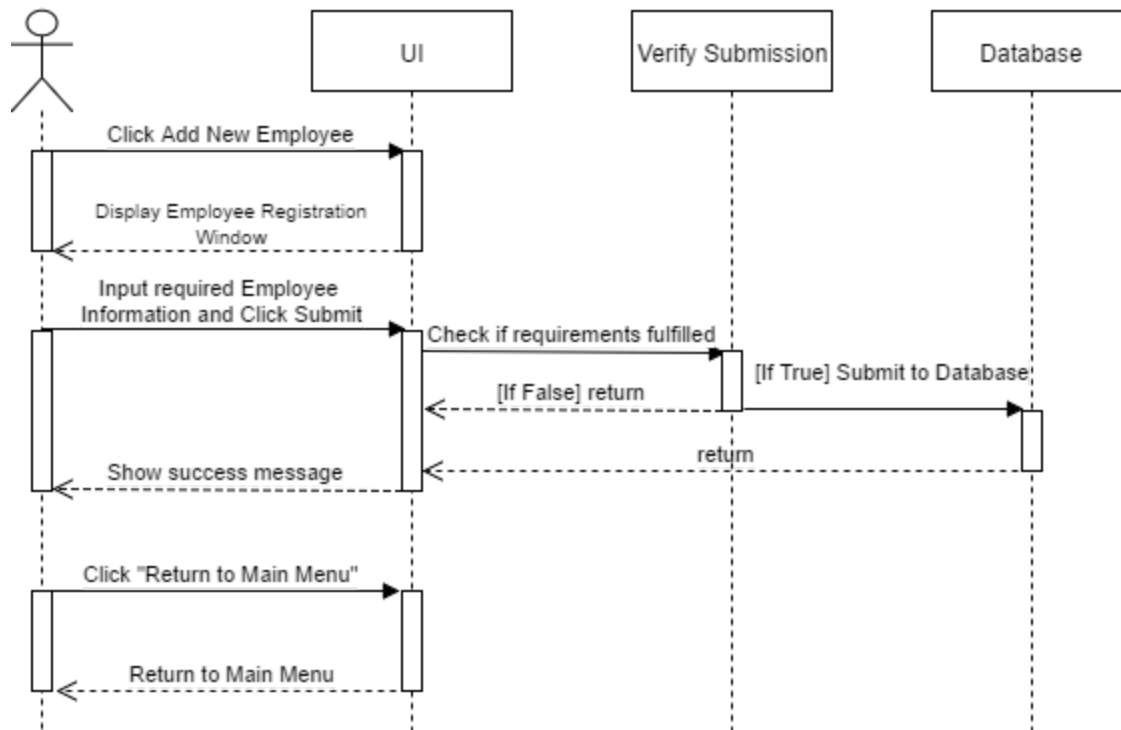
7.3.1.6 View Employee Sequence

View Employee Sequence



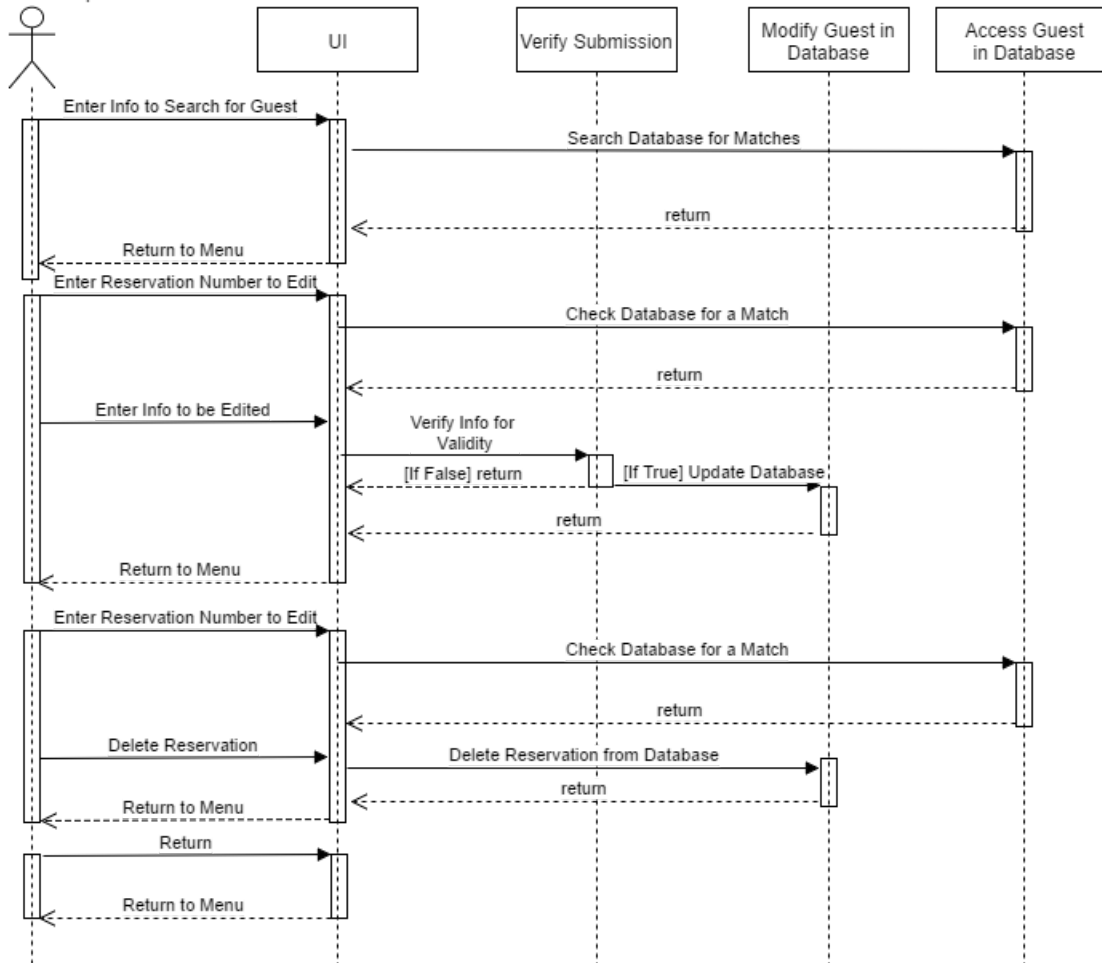
7.3.1.7 Add Employee Sequence

Add Employee Sequence



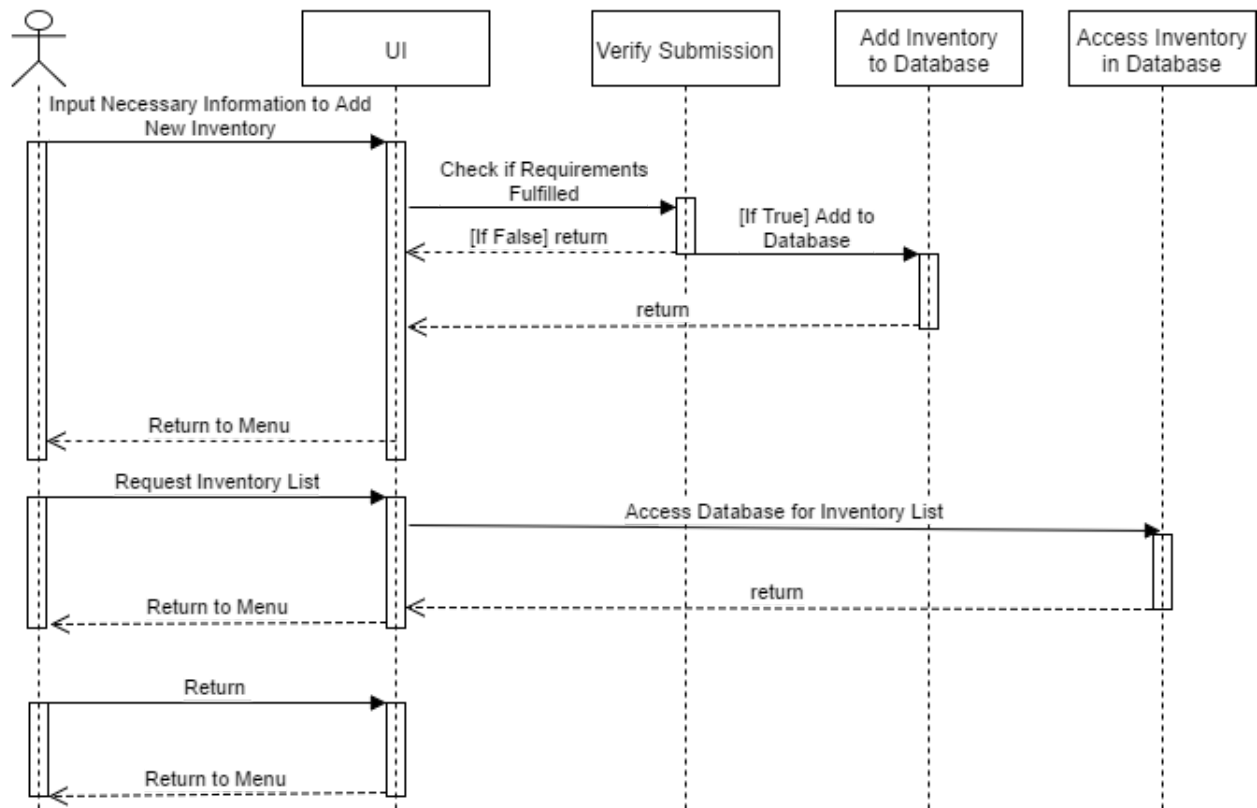
7.3.1.8 Guest Search Sequence

Guest Search Sequence



7.3.1.9 Add Inventory Sequence

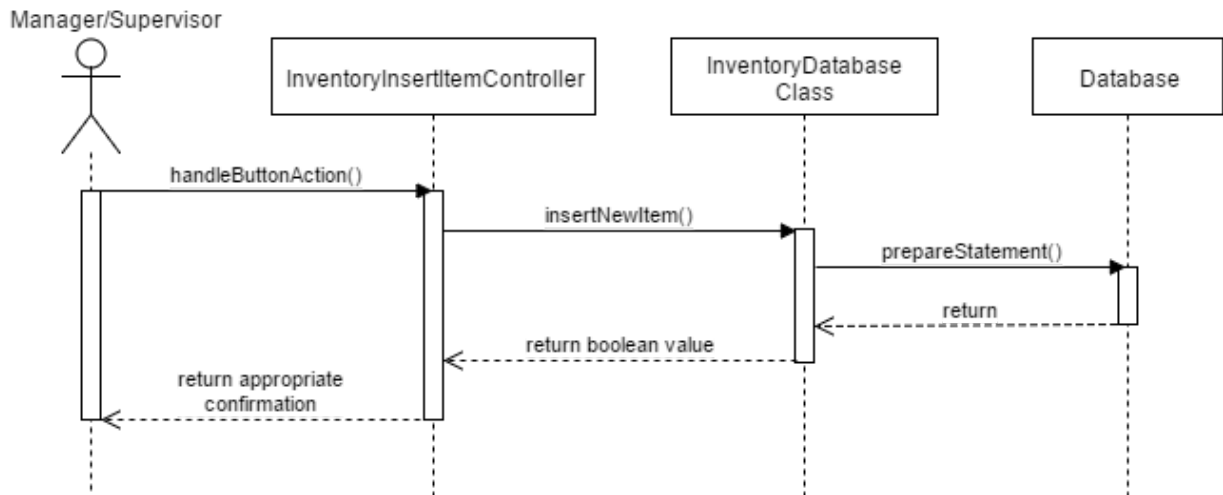
Add Inventory Sequence



7.3.2 Object Perspective Sequence Diagrams

7.3.2.1 Add Inventory

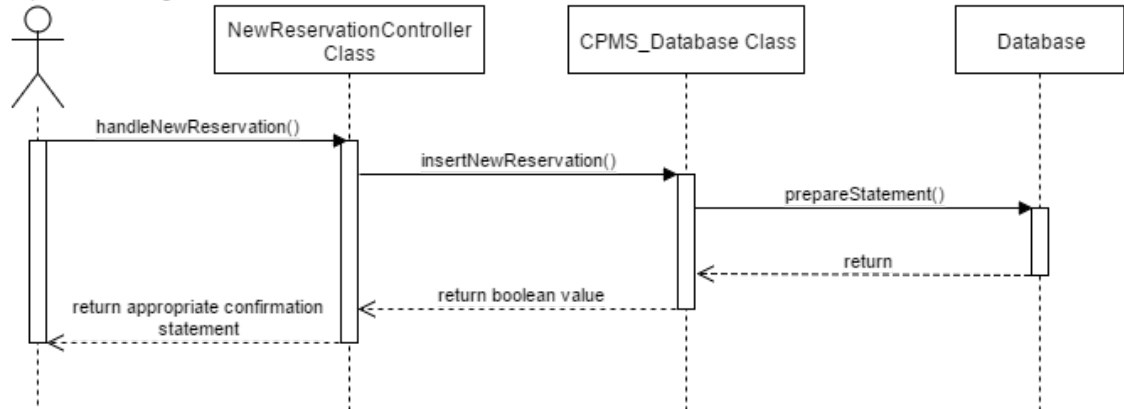
Add Inventory Object Sequence



7.3.2.2 New Reservation

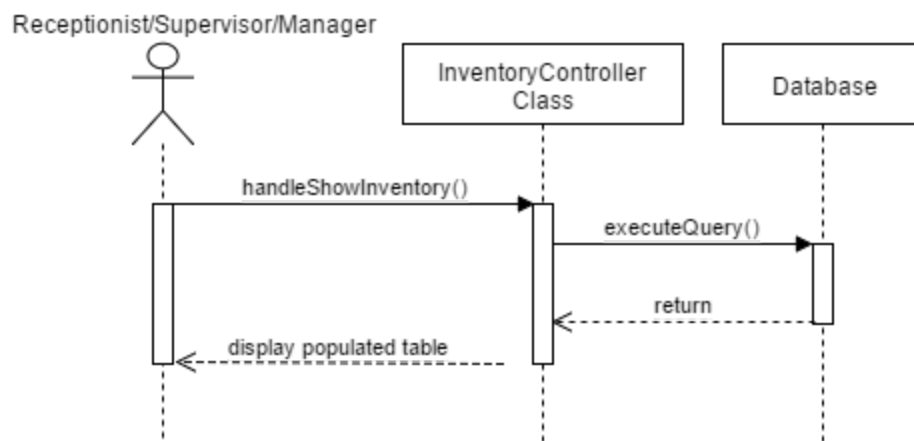
New Reservation Object Sequence

Receptionist/Supervisor/Manager



7.3.2.3 View Inventory

View Inventory Object Sequence



8. System Rationale

8.1 Architecture

The system is based on the Model View Controller (MVC) architecture design. The models are based on the object classes. Views are generated by Java FXML files. Then the controller class handles the actions from the view with the models. The approach was chosen based on the constraint of having to use Java, which encouraged the use of JavaFX/FXML for the GUI implementation. JavaFX/FXML is already an MVC based framework which made the choice between this and other less usable options an easy one. This setup closely resembles the C#/XAML environment which the team members responsible for GUI are more experienced with.

8.2 Objects

The Objects are there to reflect the actors of the system as well as the entities that the system has to interact with in order to display information to the actors and update it accordingly. This yields Employee, Manager, and Guests as objects inheriting from Human, where Manager inherits directly from Employee. This eliminates repetition of many of the same fields that “human” entities in the application domain would share in the design. Manager also has a number of methods separated to maintain the two levels of access required where only Managers can perform critical tasks that Employees shouldn’t be able to do under the given requirements. Rooms, and Items are less dynamic entities that would be updated less frequently, while Visits represent the entire stay of the guest from the moment of potential booking all the way up until a final checkout. The Room Service is a similar dynamic nature that reflects an action of service to a guest. The main object is the Crystal Palace Management System (CPMS) that provides functionalities that all users of the application will need, that have to do with the hotel operations described in the requirements elicitation.

9. Test Cases

Rationale: Please see the description for each test case for the rationale accordingly

9.1 Login Test Case

Use case name	Login
Participating actors	<ul style="list-style-type: none">• User
Description	A user inputs login information and on success will be brought to the employee main screen.
Flow of events	<ol style="list-style-type: none">1. User inputs username in the username text field.2. User inputs password in the password field.3. User double clicks login button.4. System displays employee main screen on success.
Variations	None
Entry condition	<ul style="list-style-type: none">• User enters login information.
Exit condition	<ul style="list-style-type: none">• Login information is matches stored user information and employee main screen is shown.

9.2 Add New Employee Test Case

Use case name	Add New Employee
Participating actors	<ul style="list-style-type: none">• User
Description	A user navigates to add new employee screen and fills out the required information to add a new employee to the database.
Flow of events	<ol style="list-style-type: none">1. User inputs employee ID number.2. User inputs employee first name.3. User inputs employee last name.4. User selects hire date.5. User enters employee department ID.6. User enters employee username.7. User enters temporary employee password.8. User double clicks submit button.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• Employee information is added to the database and a screen showing that the employee was added successfully is shown.

9.3 View Inventory Test Case

Use case name	View Inventory
Participating actors	<ul style="list-style-type: none">• User
Description	A user views all inventory currently in the database.
Flow of events	<ol style="list-style-type: none">1. User double clicks the show items button on the inventory screen.2. A table is displayed with all items currently in the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on the inventory screen with all inventory items currently in the database.

9.4 Add Item to Inventory test case

Use case name	Add item to inventory
Participating actors	<ul style="list-style-type: none">• User
Description	A user navigates to add inventory screen and fills out the required information.
Flow of events	<ol style="list-style-type: none">1. User enters item ID.2. User enters item name.3. User enters quantity.4. User enters description.5. User enters location
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• Item information is added to the database and a new screen is shown that the item was added successfully.

9.5 View Employees Test Case

Use case name	View Employees
Participating actors	<ul style="list-style-type: none">• User
Description	A user views all employees in the database.
Flow of events	<ol style="list-style-type: none">1. User double clicks the employees button on the view employees screen.2. A table is displayed with all employees currently in the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on the view employees screen with all employees currently in the database.

9.6 Guest Search Test Case

Use case name	Guest Search
Participating actors	<ul style="list-style-type: none">• User
Description	A user searches the database for a guest or reservation entry.
Flow of events	<ol style="list-style-type: none">1. User clicks the Guest Search button on the Employee Main screen.2. A new screen is opened with searchable text fields.3. The user enters information into at least one field.4. The user clicks the Submit button.5. A new screen opens and will display any guest or reservation information that matches the user input with the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on the Check In Found View screen with any matching guest information.

9.7 View Reservations Test Case

Use case name	View Reservations
Participating actors	<ul style="list-style-type: none">• User
Description	A user views all reservations currently stored in the database.
Flow of events	<ol style="list-style-type: none">1. User clicks the Reservations button on the Employee Main screen.2. A new screen is opened with a blank table.3. The user clicks the View Reservations button.4. The table is populated with reservation data from the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on the View Reservations screen with all current reservation information.

9.8 View Rooms Test Case

Use case name	View Rooms
Participating actors	<ul style="list-style-type: none">• User
Description	A user views all rooms with applicable information from the database.
Flow of events	<ol style="list-style-type: none">1. User clicks the Rooms button on the Employee Main screen.2. A new screen is opened with a blank table.3. The user clicks the Show Rooms button.4. The table is populated with room data from the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on Room View screen with all current room information.

9.9 View Hotel Services

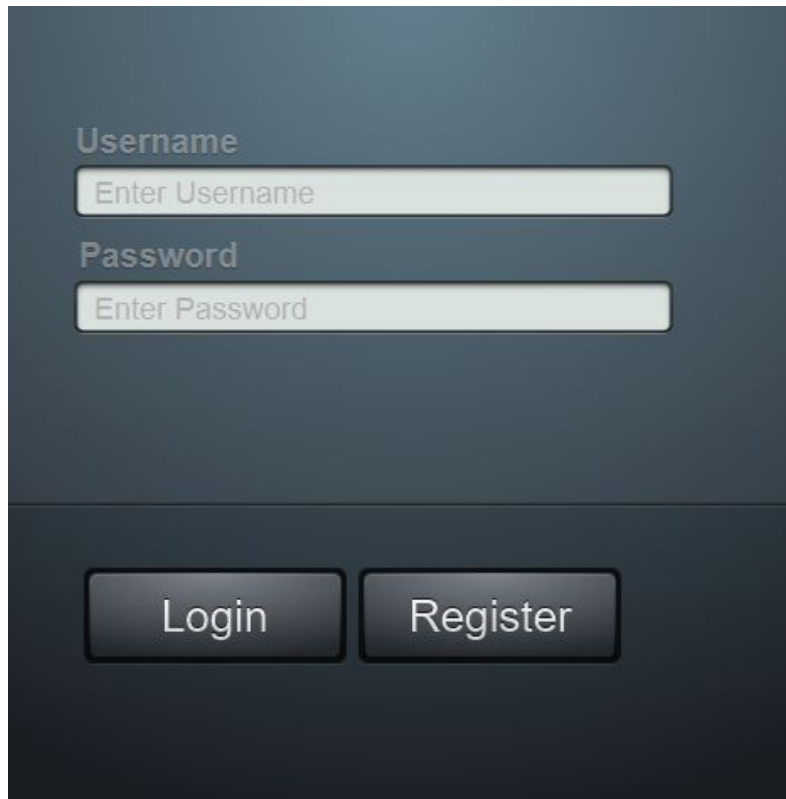
Use case name	View Hotel Services
Participating actors	<ul style="list-style-type: none">• User
Description	A user views third party service providers.
Flow of events	<ol style="list-style-type: none">1. User clicks the Hotel Services button on the Employee Main screen.2. A new screen is opened with a blank table.3. The user clicks the Show Services button.4. The table is populated with hotel services from the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on Hotel Services screen with stored hotel service information.

9.10 View Time Sheet Test Case

Use case name	View Time Sheet
Participating actors	<ul style="list-style-type: none">• User
Description	A user views time worked by employees.
Flow of events	<ol style="list-style-type: none">1. User clicks the Time Sheet button on the Employee Main screen.2. A new screen is opened with a blank table.3. The user clicks the Show Times button.4. The table is populated with employee timesheet information from the database.
Variations	None
Entry condition	<ul style="list-style-type: none">• Inherited from Login use case.
Exit condition	<ul style="list-style-type: none">• A table is populated on Time Sheet View screen with stored employee timesheet information.

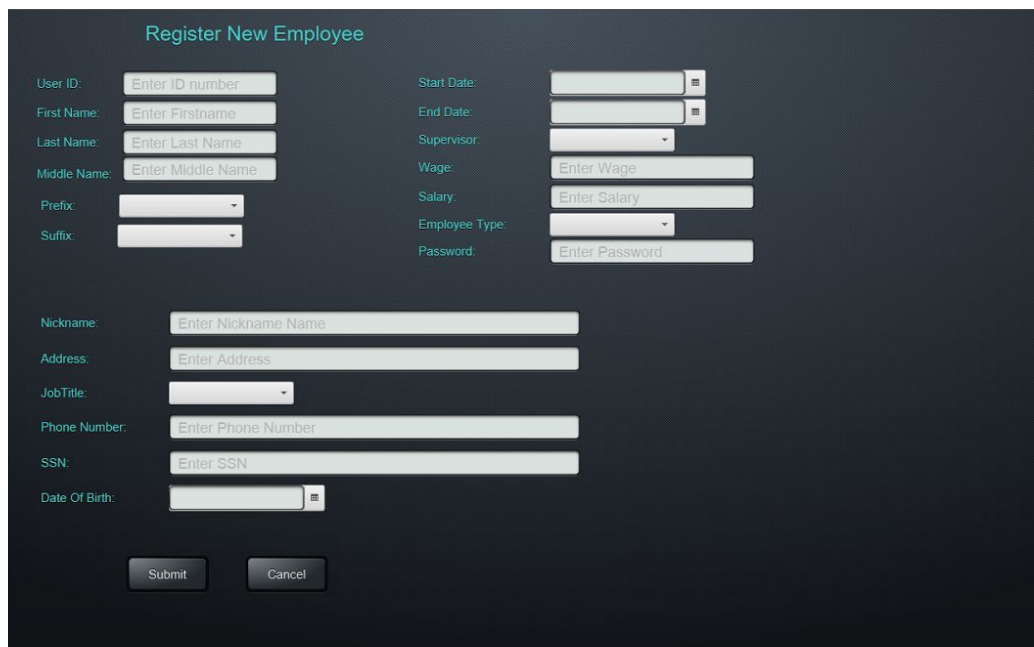
10. *Horizontal Prototype*

Login Window



A horizontal prototype of a login window. It features a dark blue background. At the top, the word "Username" is displayed in a light blue font, followed by a light blue rectangular input field containing the placeholder text "Enter Username". Below this, the word "Password" is displayed in a light blue font, followed by a light blue rectangular input field containing the placeholder text "Enter Password". At the bottom, there are two dark blue buttons with white text: "Login" on the left and "Register" on the right.

Employee Registration Window



A horizontal prototype of an employee registration window. The title "Register New Employee" is centered at the top in a light blue font. The form is organized into two columns of input fields. The left column includes: "User ID:" with a light blue input field "Enter ID number"; "First Name:" with "Enter Firstname"; "Last Name:" with "Enter Last Name"; "Middle Name:" with "Enter Middle Name"; "Prefix:" with a dropdown menu; "Suffix:" with a dropdown menu; "Nickname:" with "Enter Nickname Name"; "Address:" with "Enter Address"; "Job Title:" with a dropdown menu; "Phone Number:" with "Enter Phone Number"; "SSN:" with "Enter SSN"; and "Date Of Birth:" with a date picker. The right column includes: "Start Date:" with a date picker; "End Date:" with a date picker; "Supervisor:" with a dropdown menu; "Wage:" with "Enter Wage"; "Salary:" with "Enter Salary"; "Employee Type:" with a dropdown menu; and "Password:" with "Enter Password". At the bottom, there are two dark blue buttons with white text: "Submit" on the left and "Cancel" on the right.

Edit Employee Window

Edit Employee

User ID:	Label		
First Name:	<input type="text" value="Enter Firstname"/>	Start Date:	<input type="text"/>
Last Name:	<input type="text" value="Enter Last Name"/>	End Date:	<input type="text"/>
Middle Name:	<input type="text" value="Enter Middle Name"/>	Supervisor:	<input type="text"/>
Prefix:	<input type="text"/>	Wage:	<input type="text" value="Enter Wage"/>
Suffix:	<input type="text"/>	Salary:	<input type="text" value="Enter Salary"/>
Nickname:	<input type="text" value="Enter Nickname Name"/>	Employee Ty...	<input type="text"/>
Address:	<input type="text" value="Enter Address"/>	Password:	<input type="text" value="Enter Password"/>
JobTitle:	<input type="text"/>		
Phone Num...	<input type="text" value="Enter Phone Number"/>		
SSN:	<input type="text" value="Enter SSN"/>		
Date Of Birth:	<input type="text"/>		

Edit Reservation

Edit Reservation

*Phone Num...	<input type="text" value="Enter Phone Numbe"/>	Room Number:	<input type="text" value="Enter Room Number"/>
*First Name:	<input type="text" value="Enter Guest First Na"/>	Note:	<input type="text" value="Enter Note"/>
*Last Name:	<input type="text" value="Enter Guest Last Na"/>	Start Date:	<input type="text"/>
Middle Name:	<input type="text" value="Enter Guest Middle"/>	End Date:	<input type="text"/>
Prefix:	<input type="text"/>	Check In:	<input type="text"/>
Suffix:	<input type="text"/>	Check Out:	<input type="text"/>
Nickname:	<input type="text" value="Enter Guest Nickna"/>	Balance:	<input type="text" value="Enter Balance"/>
*Address:	<input type="text" value="Enter Guest Address"/>	Cash:	<input type="text"/>
		Total Adults:	<input type="text" value="Enter Total Adults"/>
		Total Children:	<input type="text" value="Enter Total Children"/>
		Multiple Reserv...	<input type="text" value="Enter Multiple Resen"/>

11. Cost Analysis

11.1 Function Point Cost Analysis

Measurement Parameter	Count	Simple	Weighting Factor			Complex	
			Average				
Number of User Inputs	54 x	3	4	6	= 216		
Number of User Outputs	29 x	4	5	7	= 145		
Number of User Inquiries	10 x	3	4	6	= 40		
Number of Files	30 x	7	10	15	= 300		
Number of External Interfaces	3 x	5	7	10	= 21		
Count = Total						722	

Note: By clicking on the buttons above more information about the measurement parameters will be available.

Rate each factor (F_i, i=1 to 14) on a scale of 0 to 5.

0	1	2	3	4	5
No	Incidental	Moderate	Average	Significant	Essential

Rate each factor (F_i, i=1 to 14) on a scale of 0 to 5.

F1. Does the system require reliable backup and recovery?	5
F2. Are data communications required?	0
F3. Are there distributed processing functions?	0
F4. Is performance critical?	3
F5. Will the system run in a existing, heavily utilized operational environment?	0
F6. Does the system require on-line data entry?	0
F7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	0
F8. Are the master files updated on-line?	0
F9. Are the inputs, outputs, files or inquiries complex?	2
F10. Is the internal processing complex?	2
F11. Is the code designed to be reusable?	2
F12. Are conversion and installation included in the design?	0
F13. Is the system designed for multiple installations in different organizations?	4
F14. Is the application designed to facilitate change and ease of use by the user?	5

Result. According to the input your project has: 635 FP

Function Point Cost Analysis Result: **635 FP**

11.2 Constructive Cost Model (COCOMO)

[←](#)
[→](#)
[↻](#)
[csse.usc.edu/tools/cocomoii.php](#)

(0% - 8%)

New

Reused

Modified

Software Scale Drivers

Precedentedness
 Architecture / Risk Resolution
 Process Maturity

Development Flexibility
 Team Cohesion

Software Cost Drivers

Product

Required Software Reliability
 Data Base Size
 Product Complexity
 Developed for Reusability
 Documentation Match to Lifecycle Needs

Personnel

Analyst Capability
 Programmer Capability
 Personnel Continuity
 Application Experience
 Platform Experience
 Language and Toolset Experience

Platform

Time Constraint
 Storage Constraint
 Platform Volatility

Project

Use of Software Tools
 Multisite Development
 Required Development Schedule

Maintenance

Software Labor Rates

Cost per Person-Month (Dollars)

Results

Software Development (Elaboration and Construction)

Effort = 18.9 Person-months
 Schedule = 9.7 Months
 Cost = \$0

Total Equivalent Size = 8000 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	1.1	1.2	0.9	\$0
Elaboration	4.5	3.6	1.2	\$0
Construction	14.3	6.0	2.4	\$0
Transition	2.3	1.2	1.9	\$0

Staffing Profile

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.2	0.5	1.4	0.3
Environment/CM	0.1	0.4	0.7	0.1
Requirements	0.4	0.8	1.1	0.1
Design	0.2	1.6	2.3	0.1
Implementation	0.1	0.6	4.9	0.4
Assessment	0.1	0.5	3.4	0.5
Deployment	0.0	0.1	0.4	0.7

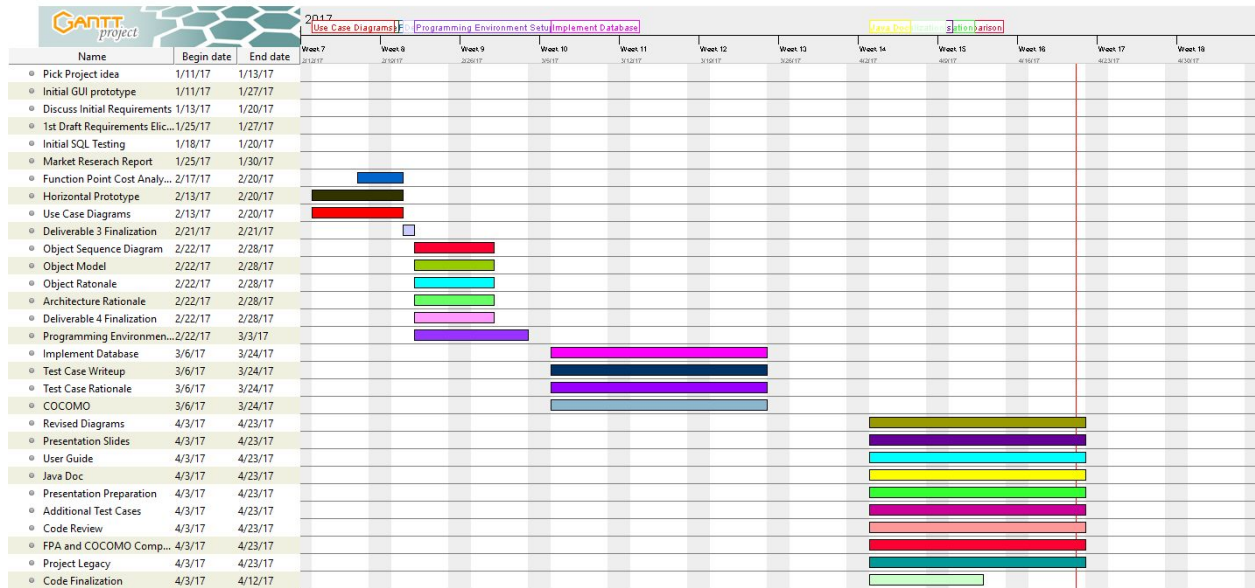
11.3 *FPA and COCOMO Analysis*

The initial Function Point Analysis yielded 635 based on the best estimates then. According to the best available formula that could be found, this converts to 17.87 person months. The COCOMO Analysis showed 18.9 person months. These figures are remarkably close even though the estimates were done several weeks apart, with the COCOMO presenting a more accurate analysis based on the work already done. This displays a good initial estimation and also reveal the amount of work necessary and speaks to the great effort exerted to make this application especially toward the latter period of the due dates.

12. Work Structure Document

Name	Role
James Gangavarapu	Documentation, UML, Requirements Elicitation
Joshua Katikala	Documentation, UML,, Requirements Elicitation, Presentation
Robert Kim	Documentation, UML, Requirements Elicitation
Eldin Mulalic	Team Coordinator, GUI, Database, Documentation,
Eden Taitt	Documentation, Market Research
Jason Tatum	GUI, Solution Architect, Database

13. Gantt Chart



14. *Dictionary of Terms*

CPMS: Crystal Palace Management System

SW: Software (requirement)

MVC: Model View Controller

15. *Current Code*

The current code is on the GitHub repository:

<https://github.com/codethismachine/CrystalPalaceManagementSystem>

16. Legacy

This project was an excellent choice due to the easily expandable requirements which could add to the many lines of code required for the project, due to the larger size of our group. Each feature would be organic to the operations of a hotel without overcomplicating the project.

Initial planning proved indispensable in spite of having to adjust many of the diagrams later on that didn't correspond to the most sensible way to implement features of this application in the final code.

The project was greatly aided by the team members' energetic and vibrant approach to steadily work on this project throughout the semester to produce an application that aligns closely with what we set out to accomplish at the beginning of this project. This project helped us adjust to the upcoming environment of collaborative efforts that we would surely face in the workplace after completion of our studies. We learned a lot in every domain involved with making this application. Most of all, we produced an application we can look back on with a great sense of accomplishment.

17. User Guide



Crystal Palace Management System (CPMS) User Guide

A screenshot of the Crystal Palace Management System (CPMS) login interface. The window has a title bar with the text "Crystal Palace Management System (CPMS) User Guide" and standard window controls (minimize, maximize, close). The background is dark gray. The login form consists of two input fields: "Username" and "Password", both with light gray borders and placeholder text "Enter Username" and "Enter Password" respectively. Below the input fields are two buttons: "Login" and "Exit", both with dark gray backgrounds and white text.

Login:

1. Enter your username and password to log in (case-sensitive)
2. Click on the Login button

A screenshot of a software window titled "Guest Search". The window has a dark gray background. At the top, there is a title bar with standard window controls (minimize, maximize, close). Below the title bar, the text "Guest Search" is displayed in a teal color. There are four input fields stacked vertically, each with a light gray border and placeholder text: "Enter Guest Last Name", "Enter Guest First Name", "Enter Guest Phone Number", and "Enter Guest Reservation Number". At the bottom of the window, there are two buttons: "Submit" and "Back", both with a dark gray background and white text.

Guest Search:

1. Click on "Guest Search"
2. Enter any desired attributes to search by
3. To display all records from this window enter the "%" character.

Edit Reservation

*Phone Num...	4046713819	Room Number:	102
*First Name:	Walter	Note:	Funny Smells
*Last Name:	White	Start Date:	4/10/2017
Middle Name:	Enter Guest Middle	End Date:	4/15/2017
Prefix:		Check In:	
Suffix:		Check Out:	
Nickname:	Heisenberg	Balance:	300.0
*Address:	308 Negra Arroyo L:	Cash:	<input type="checkbox"/>
		Total Adults:	1
		Total Children:	2
		Multiple Reserv...	Enter Multiple Resen

View/Edit Reservation and Room Service:

1. Locate the Reservation Number shown in the leftmost column and type it in the “Enter Reservation Number To Edit” box.
2. Click Edit
3. Change any desired attributes in the following “Edit Reservation” window and click Submit.
4. To view or order Room Service click on “Room Service”. This brings up a Room Service window.
5. Food can be selected from the dropdown box and a quantity entered. When this is done click Submit.
6. To cancel any existing room service order, find the “Order_ID” in the respective column and enter it in the “Enter Order ID To Cancel” box and click “Cancel Order”.

Item_ID	Item_Name	Price	Quantity	Description	Type	Location	
1	Towels Large	0.0	500	Large Towels	1	Storage	
2	Towels Medium	0.0	500	Medium Towels	1	Storage	
3	Towels Small	0.0	100	Small Towels	1	Storage	
4	Toilet Paper	0.0	100	Roll of Toilet Paper	1	Storage	
5	Soap	0.0	200	Soap packet	1	Storage	
6	Liquid Soap	0.0	50	Liquid Soap (in liters)	1	Storage	
7	Shampoo	0.0	100	Shampoo packet	1	Storage	
8	Conditioner	0.0	100	Conditioner packet	1	Storage	
9	Linens Standard	0.0	100	Linens for Standard Rooms	1	Storage	
10	Linens Deluxe	0.0	100	Linens for Standard Rooms	1	Storage	
11	Fries	1.99	100	Portions of Fries	2	Kitchen	
12	Hamburger	2.99	100	Hamburger Portions	2	Kitchen	
13	Hot Dog	1.99	100	Hot Dog Portions	2	Kitchen	
14	Coca-Cola	1.49	50	16 oz. Coca-Cola	2	Kitchen	
15	Coca-Cola Diet	1.49	100	16 oz. Diet Coke	2	Kitchen	
16	Sprite	1.49	100	16 oz. Sprite	2	Kitchen	

Show Items
Add Inventory
ReturnToMain

Show Inventory:

1. Click on the Inventory button in the main menu. Click “Show Items” to see all current items in the inventory.
2. Click “Add Inventory” to add an item to the inventory. This brings up the “Add Item To Inventory” window. This window can also be reached from the main menu.
3. Enter all corresponding attributes with 1 for non-food items and 2 for food items in the “Type”.

Register New Employee

User ID:

Start Date:

First Name:

End Date:

Last Name:

Supervisor:

Middle Name:

Wage:

Prefix:

Salary:

Suffix:

Employee Type:

Password:

Nickname:

Address:

JobTitle:

Phone Number:

SSN:

Date Of Birth:

Submit

Cancel

Add New Employee:

1. Click on Add New Employee
2. Enter all provided attributes for the employee.
3. Click Submit
4. If submission was successful, a corresponding window will appear. If any required information is missing, it will remain on this window.

The screenshot shows a web application window with a dark-themed sidebar on the left and a light-themed main content area on the right. The sidebar contains the following elements:

- A heading "Search For Employee" in teal.
- Four input fields stacked vertically: "Enter Employee Last Name", "Enter Employee First Name", "Enter Nickname", and "Enter Employee Username".
- Two buttons: "Search" and "Show All".
- A section titled "Enter Employee Username to Edit" in teal.
- An input field labeled "Enter Username".
- An "Edit" button.

The main content area is currently empty, displaying the text "No columns in table" in teal, indicating that no data is currently shown in the table.

View Employees:

1. Enter information to search by in the given search fields and click Search or click the "Show All" button to show all employees. Additionally a single "%" character will also display all employees.
2. Enter the "User_ID" of the employee and click the "Edit" button to edit this employee
3. Change any information as needed and click "Submit"

New Reservation

*Phone Number:	<input type="text" value="Enter Phone Number"/>	Room Number:	<input type="text" value="Enter Room Number"/>
*First Name:	<input type="text" value="Enter Guest First Name"/>	Note:	<input type="text" value="Enter Note"/>
*Last Name:	<input type="text" value="Enter Guest Last Name"/>	Start Date:	<input type="text" value=""/>
Middle Name:	<input type="text" value="Enter Guest Middle Name"/>	End Date:	<input type="text" value=""/>
Prefix:	<input type="text" value=""/>	Check In:	<input type="text" value=""/>
Suffix:	<input type="text" value=""/>	Check Out:	<input type="text" value=""/>
Nickname:	<input type="text" value="Enter Guest Nickname"/>	Balance:	<input type="text" value="Enter Balance"/>
*Address:	<input type="text" value="Enter Guest Address"/>	Cash:	<input type="checkbox"/>
		Total Adults:	<input type="text" value="Enter Total Adults"/>
		Total Children:	<input type="text" value="Enter Total Children"/>
		Multiple Reservation:	<input type="text" value="Enter Multiple Reservation"/>

New Reservation:

1. Click on New Reservation in the main menu.
2. Enter all known information into the resulting “New Reservation” window and click submit. A corresponding window as to a successful reservation will appear if all required information is entered correctly.

Clocking In/Clocking Out:

1. To clock in, simply click the “Clock In” button in the main menu. Only the first Clockin of the day will be recorded.
2. To clock out, click the “Clock Out” button in the main menu. *Warning: This will also sign you out of CPMS.*

<i>Phone_Number</i>	<i>Provider</i>	<i>Description</i>	
7701234567	Royal Flush Plumbing	Plumber	
7701234555	Royal Flush Plumbing	Plumber	
7701234561	Sparky	Electrician	
7701234568	Vamonos Pest Control	Exterminator	
7701234569	Jimmy McGill	Legal Counsel	
7701234571	Towelie	Linen and Toiletries Supplier	

Show Services
ReturnToMain

Hotel Services:

1. To view the set of providers on file for this hotel, click on “Hotel Services”. This will bring up a new window.
2. On the new window click “Show Services”. This will display all pertinent information to them, including the phone number to contact them by.

A screenshot of a search interface on a dark background. It features two input fields and a search button. The first input field is labeled "Search By" and contains the text "Enter ID". Below it is a second input field with the word "Or" centered between the two fields. To the right of the second input field is a dark button with the word "Search" in white text.

Time Sheet:

1. To view timesheet information, click on the “Time Sheet” button in the main menu.
2. Enter the employee’s User_ID (may also be known as username throughout this application.) to view only that employee’s entries or enter a date to view all entries for that date and click the “Search” button.

18. Resumes

James Rohan Gangavarapu

Computer Skills

- Programming Languages: Java, Python.
- Microsoft Office Suite (Word, Excel, PowerPoint), fast typing.
- Working knowledge of NetBeans, Unix/Linux scripting, Adobe Photoshop CS6, Lightroom and Bridge.
- Experienced in assembling computers from scratch.
- Learning: PHP, MySQL, C languages.

EDUCATION

- **Georgia State University**, Atlanta, GA
Expected Graduation May 2018
- Bachelor of Science, Computer Science
- Minor in Mathematics
- Overall GPA: 3.40/4.0
- CSC 2010 PRINCIPLES OF COMPUTER SCIENCE
- CSC 2310 PRIN OF COMPUTER PROGRAMMING
- CSC 2510 THEOR FOUNDATIONS OF COMP SCI
- CSC 3410 DATA STRUCTURES
- CSC 3210 COMPUTER ORG & PROGRAMMING
- CSC 3320 SYSTEM-LEVEL PROGRAMMING
- CSC 4520 DESIGN & ANALYSIS: ALGORITHMS

JOSHUA KATIKALA
Atlanta, GA
404-429-4343 | joshkatikala18@gmail.com

EDUCATION

Georgia State University, Atlanta GA
Bachelor of Science in Computer Science, May 2018
Duluth High School, Duluth GA
Diploma with Honors, May 2014 GPA: 4.0

TECHNICAL SKILLS

Java (3 years), Python, C, Assembly

PERSONAL SKILLS

Helpful, team player, diligent, detail-oriented

RELEVANT COURSEWORK

Data Structures (Java), System-Level Programming (C & Unix), Computer Organization & Programming, Theoretical Foundations of Computer Science, Principles of Computer Programming

Robert Kim
rkim5@student.gsu.edu
706-814-2083
Undergraduate at Georgia State University

Computer Science Courses Taken

- CSC 2010 - Principles of Computer Science
- CSC 2310 - Principles of Computer Programming
- CSC 2510 - Theoretical Foundations of Computer Science
- CSC 2720 - Data Structures
- CSC 3210 - Computer Organization and Programming
- CSC 3320 - System-level Programming

Additional Information

- 1.5 years of Java experience
- 0.5 years of C and SQL experience
- Experience as a sole developer of a group project. Developed a phonebook application utilizing Java and SQLite.
- Willing to learn new skills as needed for the project.

Eldin Mulalic

eldinmulalic97@gmail.com | emulalic1@student.gsu.edu | (404) 455 6860

Software Engineering Group Resume:

Skills: 2 years of Java. 1 year of C, C#, SQL, MATLAB each

Excellent research skills

GPA Overall: 3.75 **Major GPA:** 4.10

Courses taken:

Introduction to Computer Science, Principles Computer Programming, Theoretical Foundations of Computer Science, Data Structures, Math Models for Computer Science, Computer Org & Programming, System Level Programming, Computer Architecture, Digital Image Processing, Database Systems, Operating Systems, Windowing Systems Programming

In Progress:

Software Engineering, Programming Language Concepts, Design & Analysis: Algorithms, Big Data Programming

Awards:

President's List, Dean's List

EDEN K. TAITT

EDUCATION

Georgia State University – Atlanta, GA

Bachelor of Science, Computer Science, Networks and Parallel and Distributed Computing

December 2017

Minor in Spanish

- GPA: 4.03, Major GPA: 3.87
- Principles of Computer Science
- Principles of Computer Programming
- Fundamentals of Website Development
- Theoretical Foundations of Computer Science
- Computer Organization and Programming
- Data Structures
- System-Level Programming
- Computer Architecture
- Design and Analysis Algorithms

SKILLS

- Website development
- Java programming
- Microsoft Office

Awards

- C200 Scholarship for Innovative Thinking in Technology

Jason Tatum

Software Engineering Sprint 2017

January 20, 2017

Resume

Skills Programming Languages: Java, JavaFX, SQL, PHP

Technologies/Frameworks: Netbeans, Eclipse, MVVM

Relevant Courses: Data Structures, Models for Computer Science, Computer Org & Programming, System Level Programming, Introduction to Computer Science, Principles Computer Programming, Theoretical Foundations of Computer Science.

I have experience using Java to design simple to complex programs as required. GUI programming experience includes the Windowing Systems programming course offered at GSU, where I used C# and XAML to create applications. I am currently teaching myself JavaFX for this class, but the foundations are the same as using C#. Also, I have a fair amount of time creating databases using MySQL and querying such databases using a combination of SQL/PHP.