Course Name: Modern Software Development

Section: 1

Student Name: Sirisha Veeraghanta

Project Name: Hotel Reservation System

Date: November 13, 2012

Software Design Specification

Hotel Reservation System

Team members
Sirisha Veeraganta

Prepared By:
Sirisha Veeraganta

1. Introduction

This section provides a high-level overview of the design for the feature.

1.1 Purpose of the System

Hotel Reservation System software is basically used to book rooms and facilities of a hotel online. This tool helps the user to check/reserve/cancel their rooms booking. It also provides the user to view/select different categories of rooms and suites. It enables the user to view the photos of the various suites and their amenities thus promoting the hotel. The contacts page is also included in this application.

~~*Briefly describe the major aspects of the design and, if applicable, how a developer will use it. Again, be brief as this material is covered adequately in the rest of the design, For example, "Create and post a General Ledger transaction using the glTrx routines. Perform account inquiries with glJournal routines".*~~

1.2 Definitions, acronyms, and abbreviations

UML	Unified modeling language

Explain what this feature is going to deliver.

1.3 Overview of document

This document contains the architecture of the proposed solution in section 2. Section 3 contains class and object design. This section will also contain additional UML diagrams to help understand the design. Some of the UML diagrams included are class diagram and sequence diagram.

Section 4 contains the glossary defining the terms used in this document. Section 5 is the last section which contains detailed class diagrams showing attributes and methods of the classes.

1.4 References

The following references were used.

UML: http://en.wikipedia.org/wiki/Unified_Modeling_Language

2. Proposed Software Architecture

The decomposition describes the division of the feature into components and which of the requirements each component will fulfill. The intention of this section is to map the requirements from the SRS to specific component(s) and maintain basic architectural considerations (i.e. we would not expect a component that is responsible for posting to be validating a user's password). A component can be thought of as a collection of methods that provide a specific service for the feature and application. A common analogy is an object. The dependency description describes the relationships and dependencies between components.

2.1 Overview

This section describes the feature at a high level using various diagrams.

High Level Use Case Diagram

Place Use Case diagrams here

Activity Diagram

Place Use Case diagrams here

Class Diagram

Place Class diagrams here

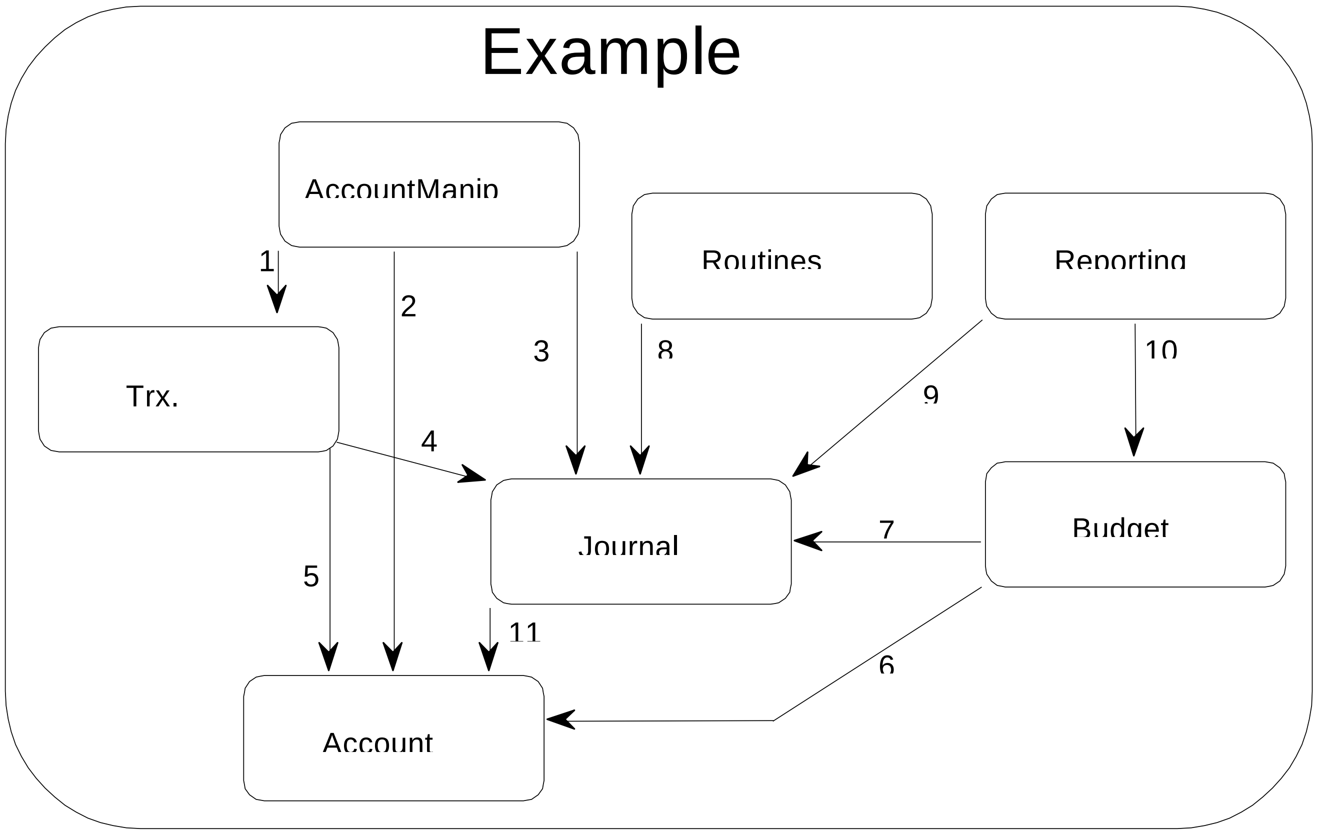
Sequence Diagram

Place Sequence diagrams here

2.2 Subsystem Decomposition

The Component diagram displays the major components of the feature. It should show dependencies between components at the highest level only. It is expected that the graphical representation will be in an OO or block diagram format.

Each Arrow should include an identifying number that should be used in section 2.1.2 that explains all dependencies for each component.



Component Dependencies

Give a brief description of each of the component links shown above.

Component	Component Description

Component (This Section should be repeated for as many Components as you have)

Brief description of the component: This should be in the form of what the component does.

Requirement Numbers

List all Requirement Numbers from the SRS for this feature that this component meets.

Req. Number	Requirement Description

Component Data

This section illustrates the logical data that makes up this component. In an OO design this may be a list of data elements that comprise the object. In a structured design this may be the logical data that this component maintains.

Pre-Conditions

List all conditions at a high level that must be met for this component to function properly.

Tables
List New or Modified Tables that this component accesses or defines.

2.3 Persistent Data Management

Data design for various persistent entities

2.3.1 Customer

Table Name	*OS/Name*	*Type*	*Security*
Table Description			
Customer	Customer	Regular	High security
This table contains information about customer, their login Id's, passwords and other personal information such as name, email, phone and zip code.			

Field Name(s)	*Physical Name*	*DataType*
Field Description		
FirstName	FirstName	Varchar(50)
This field contains first name of a customer. The field length is 50 characters.		
LastName	LastName	Varchar(50)
This field contains last name of a customer. The field length is 50 characters.		
UserName	UserName	Varchar(50)
This field contains last name of a customer. The field length is 50 characters.		
Password	Password	Varchar(50)
This field contains password of a customer. The field length is 50 characters.		
Phone	Phone	Varchar(10)
This field contains phone of a customer. The field length is 10 characters.		
Email	Email	Varchar(50)
This field contains email of a customer. The field length is 50 characters.		

Index Name(s)	*Primary*	*Modify*	*Clustered*	*Unique*	*Fields*
Index Description					
UserNameIdx	Yes	No	No	Yes	UserName
This index is based on the user names of the customers. This index guarantees that customers will always have unique user names.					

2.3.2 HotelRooms

Table Name	*OS/Name*	*Type*	*Security*
Table Description			
HotelRooms	HotelRooms	Regular	Med security
This table contains information about hotel rooms including type, rate and maximum number of rooms for each type of a room type			

Field Name(s)	*Physical Name*	*DataType*
Field Description		
RoomType	RoomType	Varchar(50)
This field contains the room type information of rooms in a hotel. This can include Deluxe, Super Deluxe and Super Deluxe with Pool.		

RoomRate	RoomRate				decimal
This field contains the cost of each room					
MaxAvailable	MaxAvailable				int
This field contains the maximum number of rooms the hotel has for each room type					
Index Name(s)	*Primary*	*Modify*	*Clustered*	*Unique*	*Fields*
Index Description					
RoomTypeIdx	Yes	No	No	Yes	RoomType
This index is based on the room type of the room in a hotel.					

2.3.3 RoomReservation

Table Name		*OS/Name*		*Type*	*Security*
Table Description					
RoomReservation		RoomReservation		Regular	High security
This table contains information about hotel room reservation made by customers for a particular date					
Field Name(s)	*Physical Name*				*DataType*
Field Description					
UserName	UserName				Varchar(50)
This is an identifier for a customer who made the reservation					
RoomType	RoomType				Varchar(50)
This is an identifier for the room type.					
CheckinDate	CheckinDate				Date
The checkin date for the reservation					
CheckoutDate	CheckoutDate				Date
The checkout date for the reservation					
This field contains the room type information of rooms in a hotel. This can include Deluxe, Super Deluxe and Super Deluxe with Pool.					
NumberOfRooms	NumberOfRooms				Int
The total number of rooms reserved in a reservation					
Index Name(s)	*Primary*	*Modify*	*Clustered*	*Unique*	*Fields*
Index Description					
RoomReservationIdx	Yes	No	No	Yes	• UserName • RoomType • CheckinDate • CheckoutDate
This index is based on the user name, room type, check in and check out dates of a reservation.					

2.4 Activity Diagram

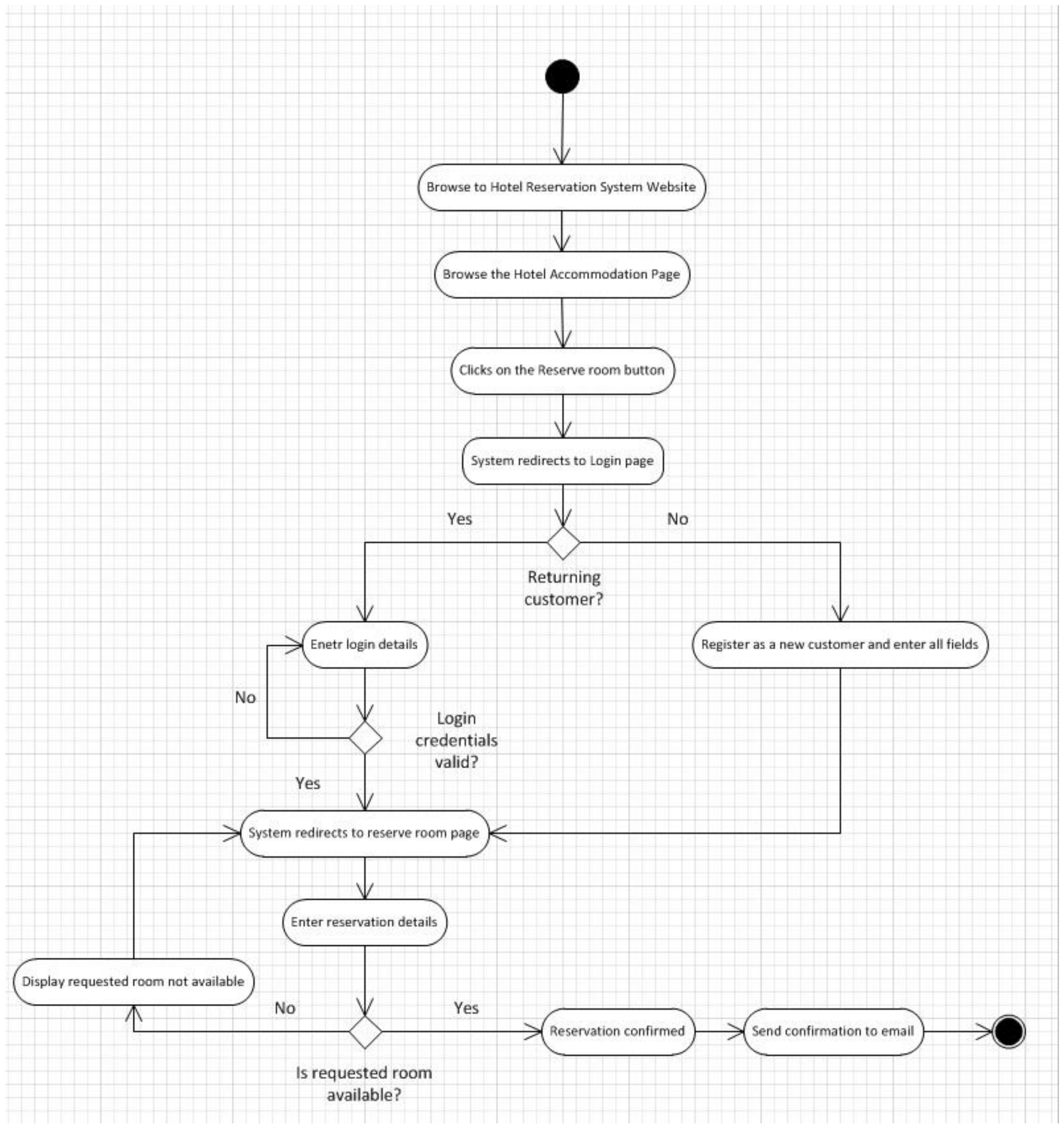

Browse to Hotel Reservation System Website
Browse the Hotel Accommodation Page
Clicks on the Reserve room button
System redirects to Login page
Yes
No
Returning customer?
Enetr login details
Register as a new customer and enter all fields
No
Login credentials valid?
Yes
System redirects to reserve room page
Enter reservation details
Display requested room not available
No
Yes
Reservation confirmed
Send confirmation to email
Is requested room available?

Figure 1: Activity diagram for Hotel Reservation System

3. Class & Object Design

This section should document at the highest level all entry points for external applications. This includes all interfaces for other features/components as well as interfaces for non-native applications (such as interfaces to Stored Procedures…)

3.1 Detailed Class Design

- Customer class
    - This class contains information about a customer which includes First Name, Last Name, User Name and other details. This class is persistent class meaning the class exposes methods to search the database and also use the class to save information to the data base. More details in appendix A and B
- Room class
    - This class contains information about a room which includes Room type, room and room rate. This class is persistent class meaning the class exposes methods to search the database and also use the class to save information to the data base. More details in appendix A and B
- Reservation class
    - This class contains information about customer's reservation which includes user name of the customer who made the reservation, room type in the reservation and check-in and checkout dates. This class is persistent class meaning the class exposes methods to search the database and also use the class to save information to the data base. More details in appendix A and B
- UI Helper class
    - A helper class which handles the requests from the customer and invokes the appropriate class to take further action
- LoginPage class
    - This class represents the login web page. It gets called when the user clicks the login button on the login page. It delegates method calls to UIHelper class for further action
- RegisterPage class
    - This class represents the register web page. It gets called when the user clicks the register button on the register page. It delegates method calls to UIHelper class for further action
- ReservationPage class
    - This class represents the reservation web page. It gets called when the user clicks the reserve button on the reservation page. It delegates method calls to UIHelper class for further action
- CancelReservationPage class
    - This class represents the cancel reservation web page. It gets called when the user clicks the cancel reservation button on the cancel reservation page. It delegates method calls to UIHelper class for further action
-

3.2 Object Interaction

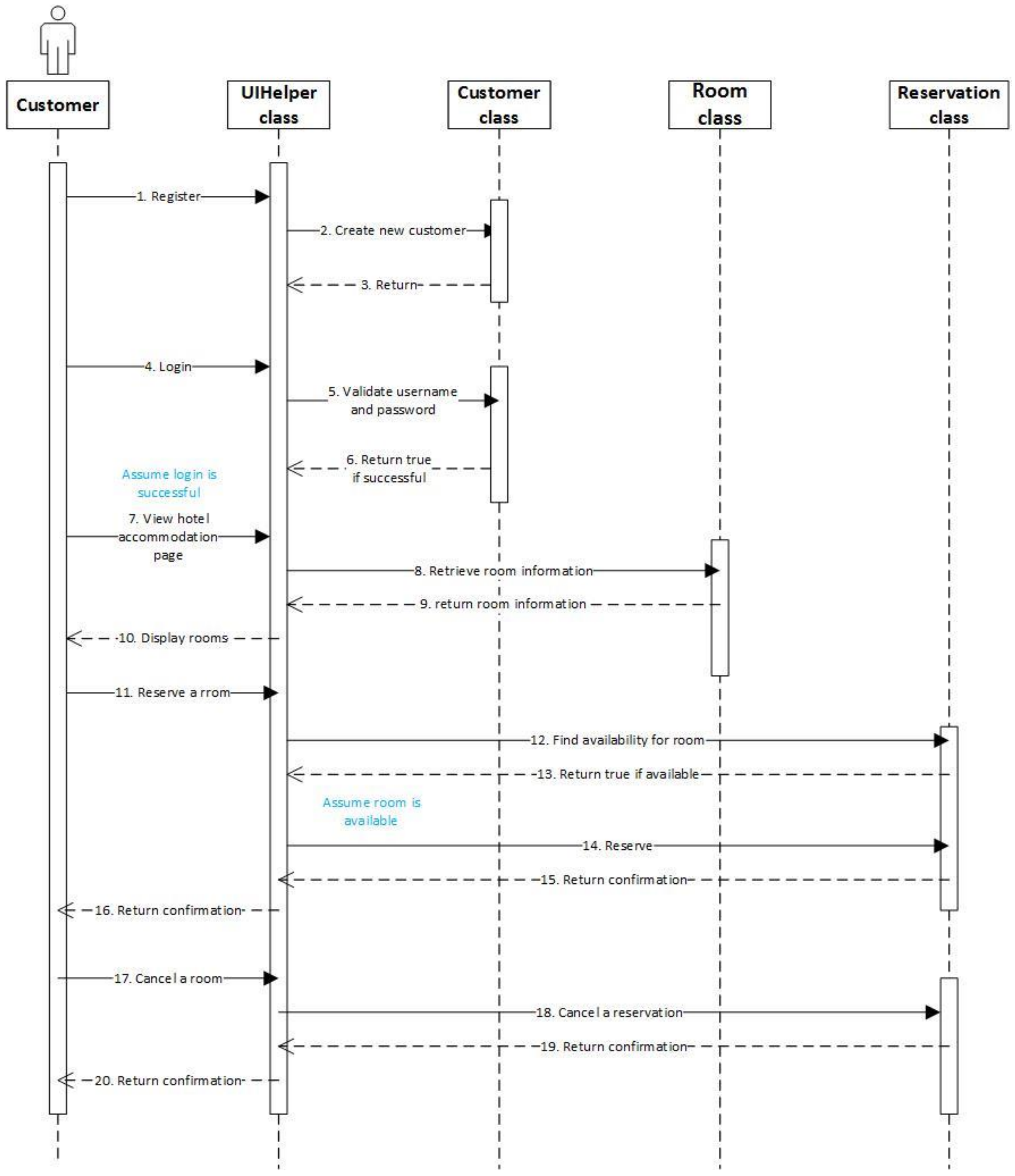


Figure 2: Sequence diagram for the Hotel Reservation System

4. Glossary

5. Appendix

5.1 Appendix A: Class Diagrams

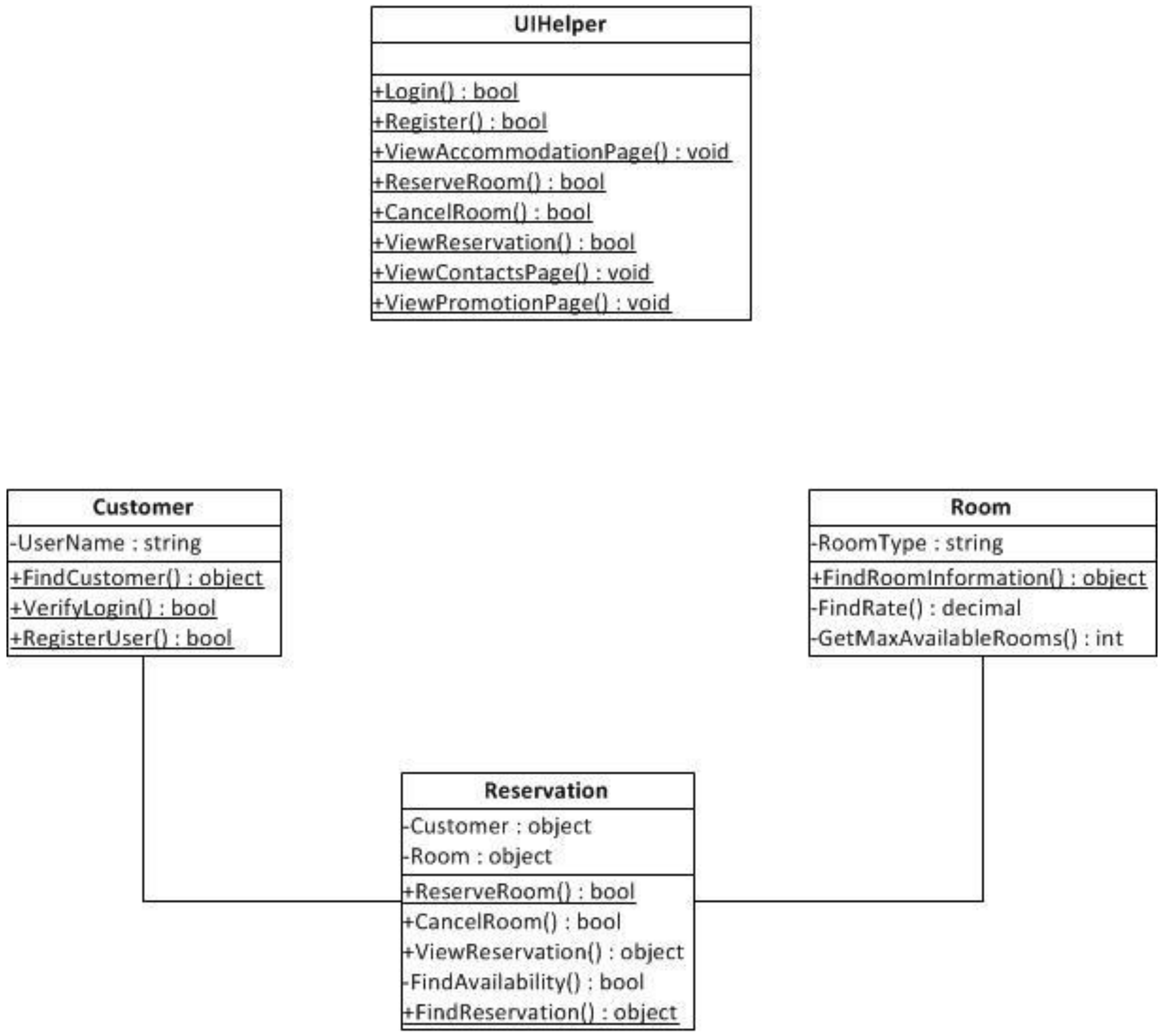


Figure 3: Class diagram for Hotel Reservation System

5.2 Appendix B: Class details

Data Requirements

1) List all of the requirements of this class. Requirements define limits and ranges, assumptions, validation rules, limitations on users of this data.

Class Invariant

1) Used to indicate the relationships between data as the state of the object changes.
   For example: PO PD object
   Status = New and LastPrintedDate = undefined and LastEditedDate = undefined

Data Structures

Variable Name	Data Type	Description and Constraints

Constants

Define any constants that will be used in the following method definitions. Note that the Constant name must follow any conventions outlined in the project team's coding standard.

Constant	Value	Description

Conversion

1) List all considerations that should be made when changing an existing file/object. For example, when adding new fields to a file, specify the field value for existing records.

7.3 Class (Static) Methods

These methods are operating on an entire collection as opposed to one object. They are considered helper routines for an object. They do not require an instance of the object to exist, do not take a parameter of Me and should not reference Me.

MethodName1 (Repeat this for all static methods in the class)

Description: Method Description - should not describe their implementation, just their purpose. (e.g. GetNextNumber - Get the next available number vs. ReadNextNumber - Read the table to get the next number; increment the number and write it to the table)

Preconditions: Document assumptions for this method, primarily parameter and state verification. (e.g. range must be set-up, parameter must be valid} If there are not assumptions then note "None".

Post Conditions: Checks that see if the routine did what it was supposed to. (e.g. Create method would verify that Me**.**IsCreated = true)

Return Type: List the return type of the method.

Interface:

Parameter	Data Type	Description

Implementation:

```
Give the implementation of the method here. e.g.

static int doubleResult(int input)
{
   return int * 2;
}
```

7.4 Public Methods

These methods are available to all users of this object, both inside and outside this module.

MethodName1 (Repeat this for all static methods in the class)

Description: Method Description - should not describe their implementation, just their purpose. (e.g. GetNextNumber - Get the next available number vs. ReadNextNumber - Read the table to get the next number; increment the number and write it to the table)

Preconditions: Document assumptions for this method, primarily parameter and state verification. (e.g. range must be set-up, parameter must be valid} If there are not assumptions then note "None".

Post Conditions: Checks that see if the routine did what it was supposed to. (e.g. Create method would verify that Me**.**IsCreated = true)

Return Type: List the return type of the method.

Interface:

Parameter	Data Type	Description

Implementation:

```
Give the implementation of the method here. e.g.

static int doubleResult(int input)
{
   return int * 2;
}
```

7.5 Restricted Methods

These are methods that for various reasons should not be used by the user of an object except in certain rare circumstances. For example, these methods may directly get and set the internal state of the object, exposing implementation details that may change (such as the Next Document Number); they may be methods on a child object that are used by its parent and should only be accessed through the parent so that the parent works correctly; they may bypass error checking for performance reasons in cases where the data is known to be valid (to copy an object, for example). **There is a reason these methods are restricted! Use them with caution or not at all!**

The description of each restricted method must indicate why it is restricted.

MethodName1 (Repeat this for all static methods in the class)

Description: Method Description - should not describe their implementation, just their purpose. (e.g. GetNextNumber - Get the next available number vs. ReadNextNumber - Read the table to get the next number; increment the number and write it to the table)

Preconditions: Document assumptions for this method, primarily parameter and state verification. (e.g. range must be set-up, parameter must be valid} If there are not assumptions then note “None”.

Post Conditions: Checks that see if the routine did what it was supposed to. (e.g. Create method would verify that Me**.**IsCreated = true)

Return Type: List the return type of the method.

Interface:

Parameter	Data Type	Description

Implementation:

```
Give the implementation of the method here. e.g.

static int doubleResult(int input)
{
   return int * 2;
}
```

7.6 Private Methods

These methods are only available to other methods within this object.

MethodName1 (Repeat this for all static methods in the class)

Description: Method Description - should not describe their implementation, just their purpose. (e.g. GetNextNumber - Get the next available number vs. ReadNextNumber - Read the table to get the next number; increment the number and write it to the table)

Preconditions: Document assumptions for this method, primarily parameter and state verification. (e.g. range must be set-up, parameter must be valid} If there are not assumptions then note “None”.

Post Conditions: Checks that see if the routine did what it was supposed to. (e.g. Create method would verify that Me**.**IsCreated = true)

Return Type: List the return type of the method.

Interface:

Parameter	Data Type	Description

Implementation:

```
Give the implementation of the method here. e.g.

static int doubleResult(int input)
{
   return int * 2;
```

9. Requirements Tracing

This section should describe the mapping of requirements to design components

Component	**Requirements satisfied by this Component**
A	Req. ID 100, 104, 105
B	Req. ID 102, 103, 112
C	Req. ID 106, 107, 108

10. Revision History

Status Key **DR :** Draft **IR:** In Review **AP:** Approved **RW:** Rework	**Revision History Of This Document** When this document requires update, document the revision details below and notify affected parties.			
Date	**Author/ Updater**	**Status**	**Item Changed**	**Short Description of Change**
11/13/12	Sirisha	Started	Added	Created initial document.