DEPARTMENT OF INFORMATION SYSTEMS

SYSTEMS DEVELOPMENT B

SYSTEMS SPECIFICATION FOR ERICA HOTEL RESERVATION

TEAM MEMBERS

Student Number: ATWMAR001	***Student Name: ATWEBEMBIRE MARTIN***
Student Number: NKNNCE003	***Student Name :NCEDILE NKONYANA***

TABLE OF CONTENTS

1. INTRODUCTION

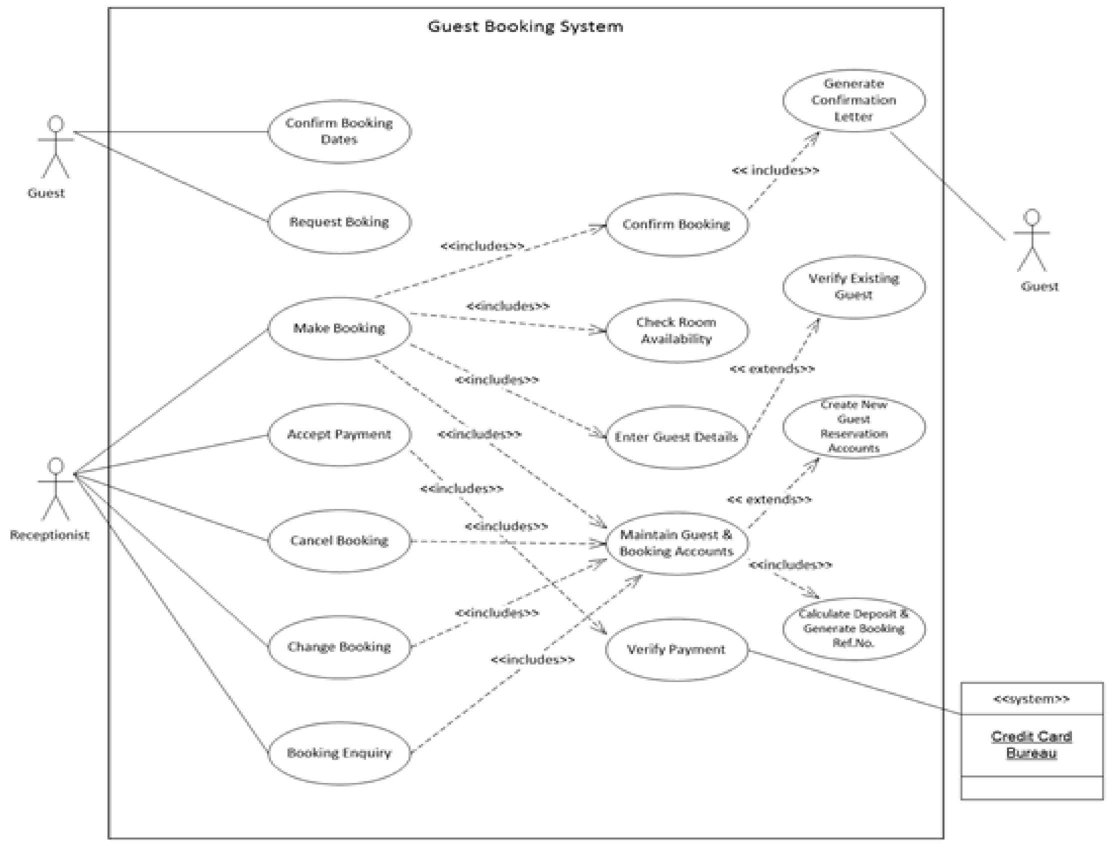
1.1. OVERVIEW OF SPECIFICATION

This document is produced as a description of the Erica Hotel Reservation system design and illustrates the functional and non – functional requirements of the aforementioned system through the definition of its architecture, data, components and interfaces.

*This **system design** document is arranged in such a way as to fully cover several major categories including outputs and controls, database design, class and sequence diagram design, dialogue design, test and implementation plan. In incense we are currently in the **system design phase** documenting the physical design of the Erica Hotel Reservation system and in doing so realizing its logical design as discussed in the analysis phase.*

*Given that we are building **a segment** of the reservation system that depicts **only** the **Guest Booking** function as previously discussed in the completed **analysis phase**, we will structure this document in such a way as to fully realize only the functionality depicted in the guest booking use case diagram below.*

Fig.1



It is important to note the various changes that have occurred since the analysis phase. Since the guest booking system is only a segment of the reservation system, majority of the functions such as handling debtors, room inventory and many others discussed in the analysis phase will not be implemented. Finally, in documenting the design phase of the

Erica Hotel Reservation system we will be looking at user interface and dialogue design, sequence, state machine and class diagram design, report design, entity relationship diagrams, Input - Output standards and controls and last but not least implementation and test plans.

1.2. Context & Scope of System Specification

In an effort to create a standardised business practice across its hotels through implementing a new standard computer software system, Erica Hotel Group had hoped to standardize its customer accounts, financial reports, client-facing documents among many others. In the design phase we are tasked with standardizing hotel reservations by computerizing the client reservation process outline in the analysis phase.

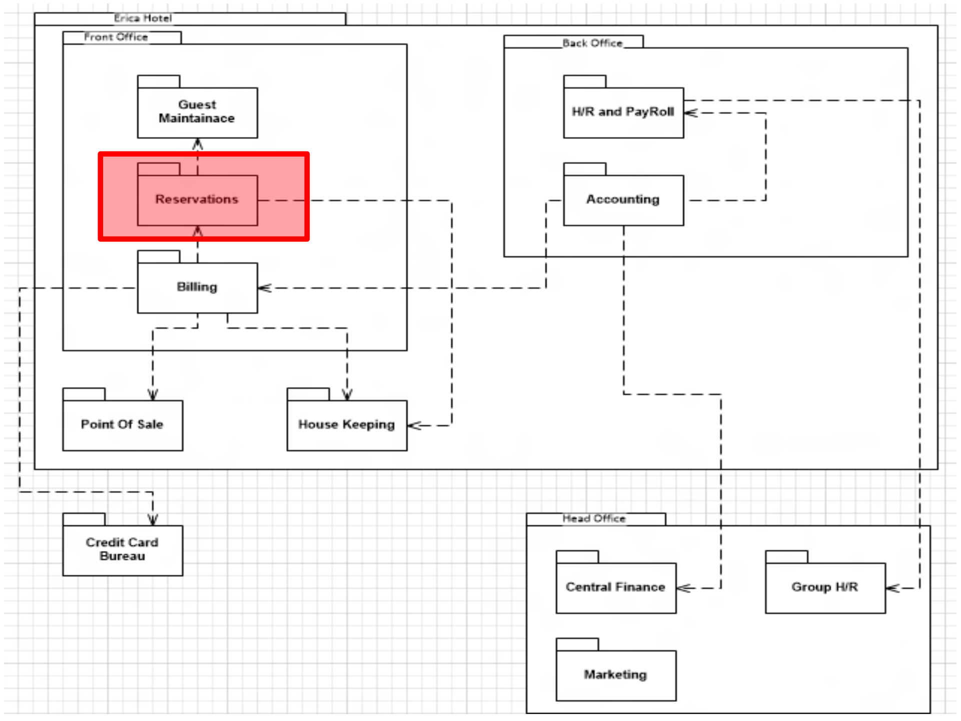
"The receptionist at the hotel is in charge of making reservations for clients. She begins by collecting the (potential) client's information; after attaining how many rooms need to be booked and over which period she checks the availability of rooms over that period and if possible she will make the reservation. Clients are required to provide a credit card number when making a reservation, alternatively they make deposit; especially during long stays over peak season. It is the receptionist's job to ensure that clients provide these requirements. It is also her job to handle cancellations and changes to bookings."

In computerizing this process we hope to eliminate several **problems** such as **reservation errors, accounting errors** and thereby give way to various **opportunities** such as implementation of a more sophisticated and efficient reservation and billing system, better creditor and room occupancy management among many others.

By applying an in-house development approach chosen in the analysis phase to design the Erica Hotel Booking System we will obtain a stepping stone to accomplish several **SMART objectives** discussed fully in the analysis phase such as establishing: - a centralised hotel management system, a secure up-to-date computer system, significant fraud reduction and automating redundant tasks and process .

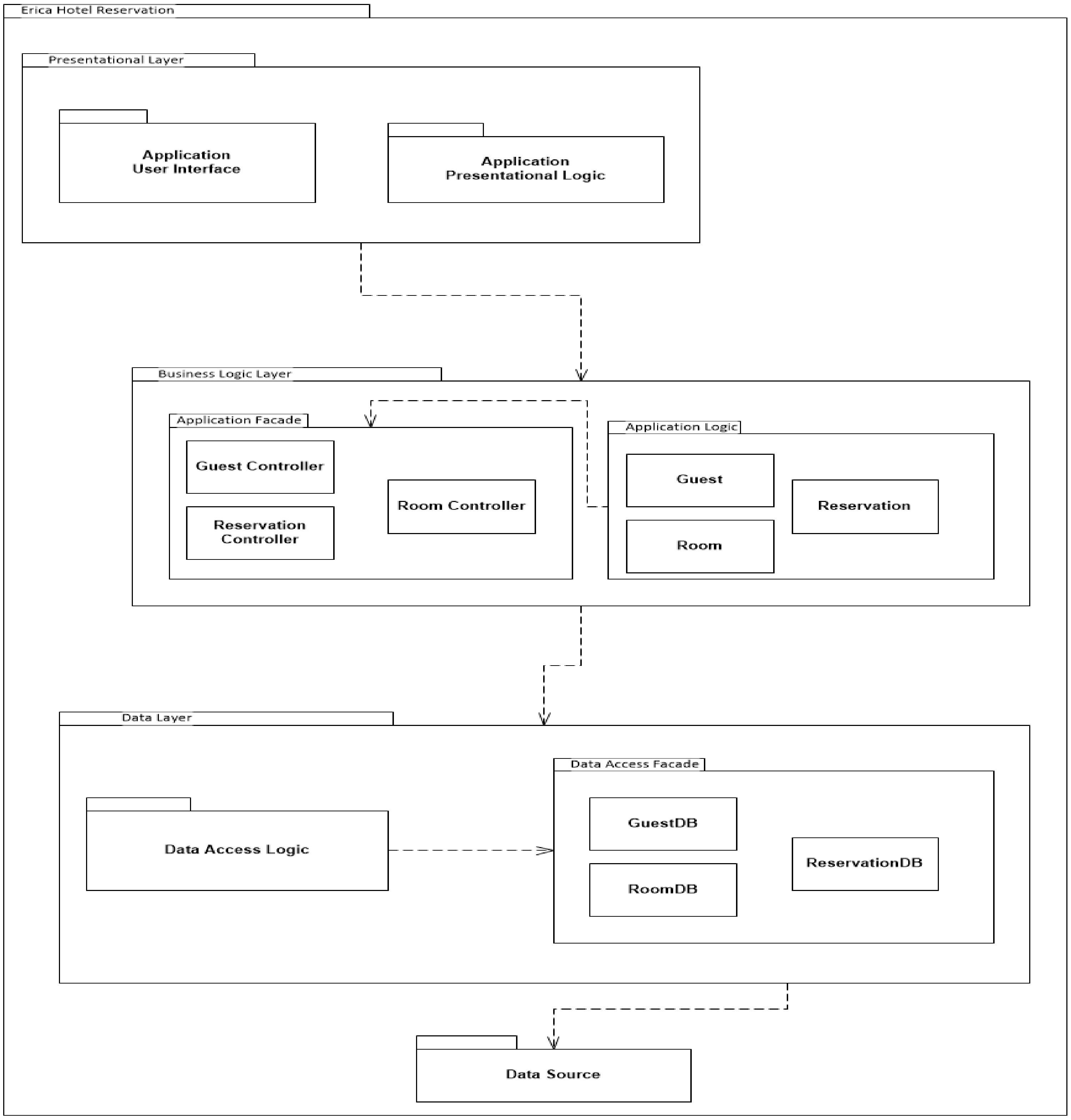
In the analysis phase we described the scope of the entire Erica Hotel System as illustrated in the package diagram below.

Fig.2



The particular part of the system being developed is highlighted and its scope is fully illustrated below in a 3 – tier architectural format depicting all namespaces/ package dependences. **One or two sample classes** are shown in their corresponding packages/namespaces. To simplify access to the various sub-systems all interactions are routed through the **facade** package which consists of controller classes.

Fig.3



Statement of Scope

In scope:

Make Guest Booking, including

- Collect and Enter Customer Details into system
- Make and Confirm Reservations
- Check room availability
- Customer billing

Change a Guest Booking, including

- Change Existing booking
- Adjust Occupancy details accordingly

Cancel a Guest Booking, including

- Cancel current booking
- Adjust Occupancy details accordingly

Make a Guest Booking Enquiry

- Enquire and Review guest booking
- Determine Booking Status

Out of scope:

Authorize deposits paid at time of booking

Given the Erica Hotels computerization project is meant to be completed in a number of iterations, this particular **scope** has been chosen so as to complete only a **single iteration** to construct and implement the required **Guest Booking** function.

1.3. DESIGN ASSUMPTIONS & CONSTRAINTS

In designing the Guest Booking system we are bound to encounter restrictions on the conditions under which the system is developed. These limitations on the design of the new system referred to as design constraints include:

- **Restriction of design options** such as limiting the system development technologies to Visual C# and MS SQL 2008 will lead to a loss of development freedom.
- **Restriction of time taken to develop system** such as having a one month developmental period.
- **Restriction of system compatibility and standard** such as system being compatible with existing system maintaining standardized outlook and allowing for easy data migration and integration with other hotels in the group.

Additionally, there are constraints imposed on the system design by non – functional requirements and these are greatly focused on system performance and quality leading to **design trade-offs** such as resources **versus** productivity and can be evaluated using the factors outlined below:

- **Expandability** which measures the designed system's ability to efficiently accommodate increased processing requirements will require the designed system to accommodate increased input-output data capacity growth by allowing for higher speeds and a dynamic memory.
- **Compatibility and reliability** allowing for the designed system to operate adequately at any given time and interface efficiently with other computer systems.
- **Maintainability and adaptability**

In developing the system design we will make use of several **assumptions** some of which are outlined below:

- New developed system will be run on Windows Operating System XP SP1 or later version running MS SQL 2008 Server.
- All deposits have been authorized therefore system design will not cater for payment authorization.
- All reservations are made by an individual from her on referred to as guest and not by an agent or third party.

2. USER INTERFACE & DIALOGUE DESIGN

In this section of the design phase we are looking at the part of the system with which the user interacts. This includes the display screens, data capture forms, generated reports, etc.

This particular section is documented through user navigation design which looks at how the user navigates/interacts with the various graphical software interfaces, input design which looks at all inputted data and their corresponding characteristics such as range, type, length, mandatory and optional fields, data entry controls, input forms and fields, access and security restrictions and output design which looks at the display forms , messages associated with inputs such as error messages and the reports produced by the system.

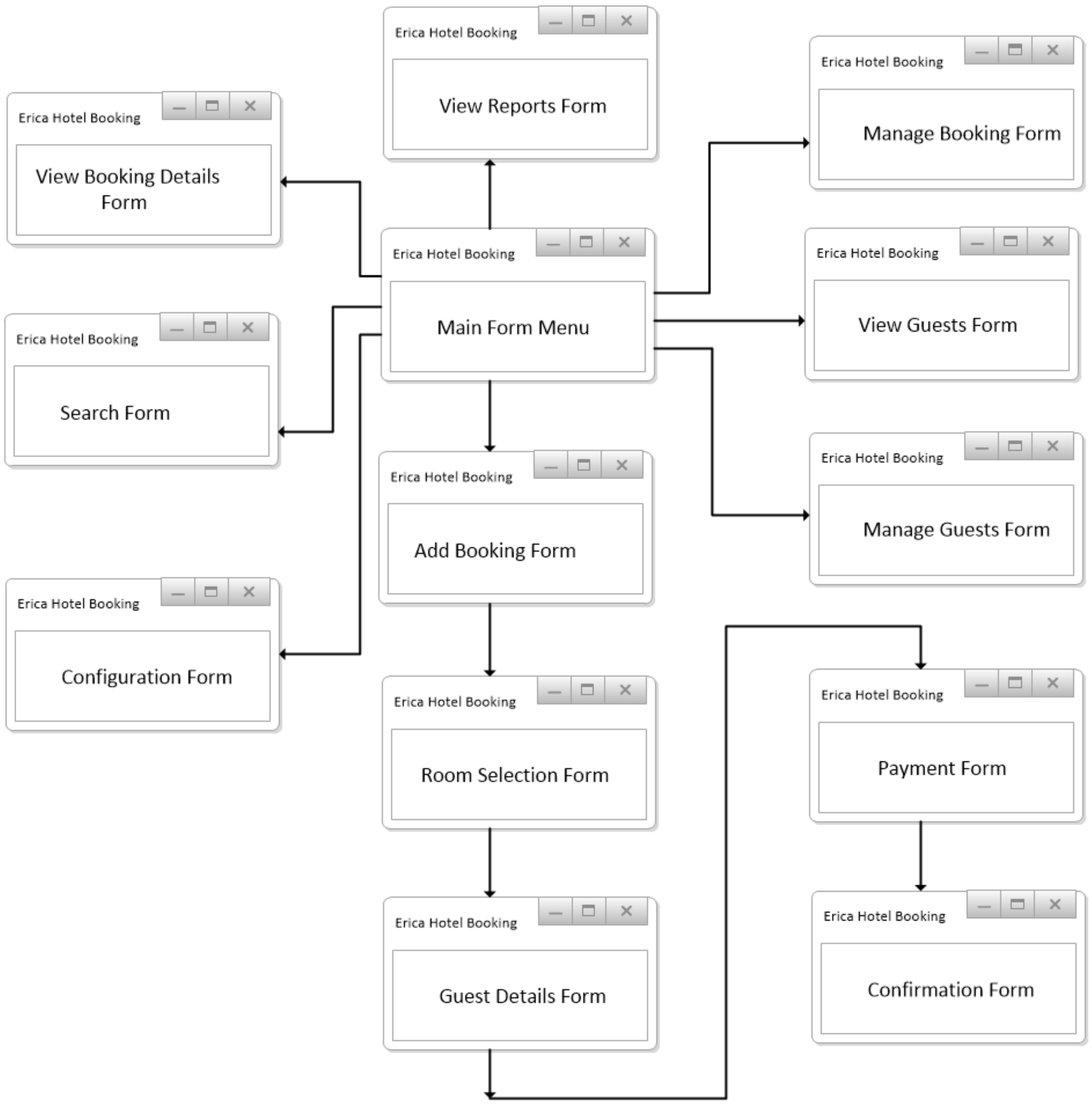
Documenting user interaction will require the use of a wire frame diagram which focuses on conveying the general structure and content requirements of the given user interface and/or a windows navigation diagram which documents the structure of the user interface by defining interaction of the basic system components, component states and corresponding transition paths. By adding realistic details to the obtained wire frame diagram and applying specific screen design principles and standards we can obtain a detailed design of each individual screen as categorically illustrated below.

2.1. WIREFRAME DIAGRAM OR WINDOWS NAVIGATION DOCUMENT

In order to model the dialog between the user and the system, we will first provide a hierarchical view of the **entire system structure** and this will be done using a **wireframe diagram** as this type of diagram is more suitable to provide a blue print of the Erica Hotel

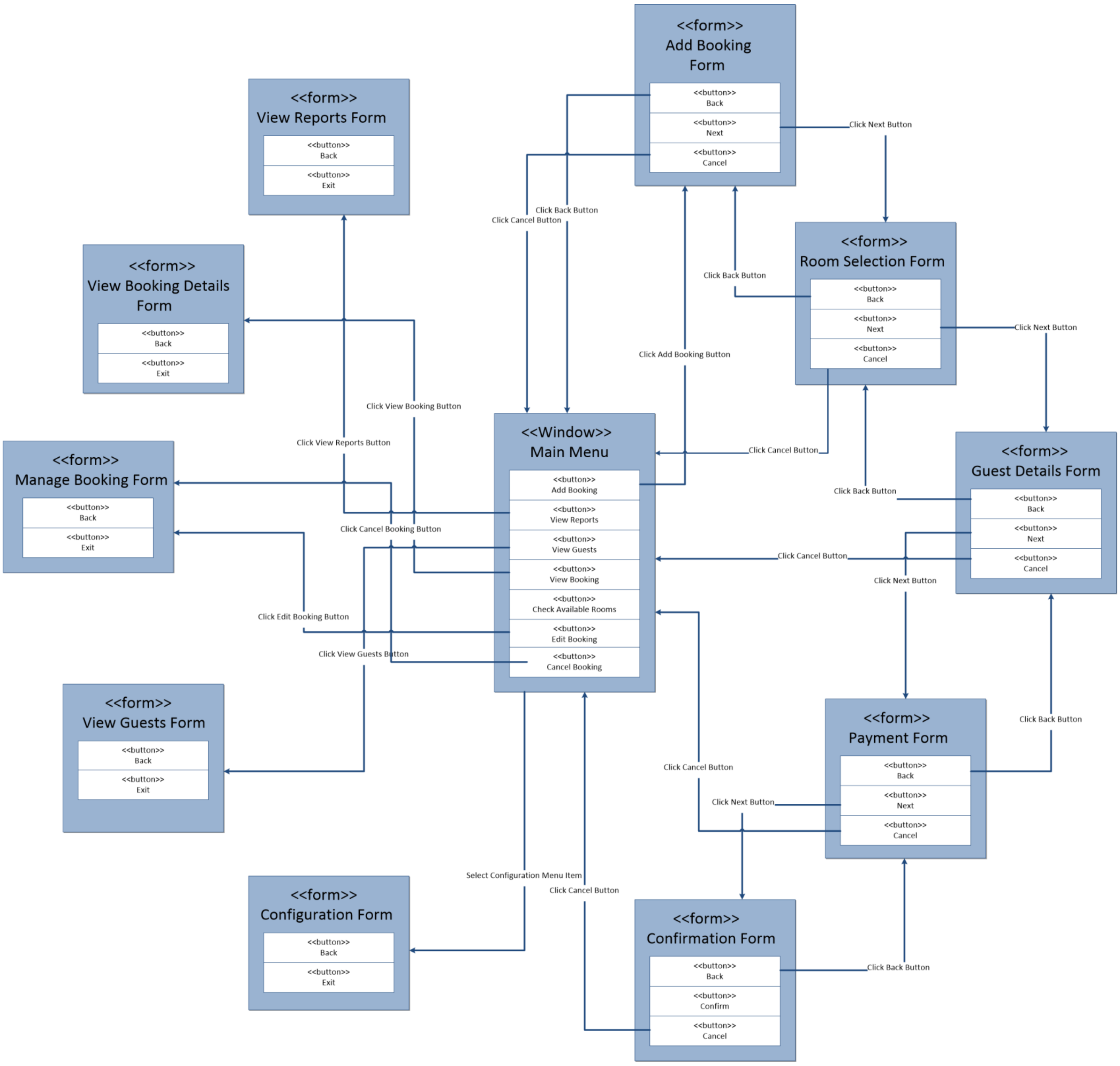
Reservation system illustrating important content and functionality. Shown below is a wireframe diagram of a high level view of the system structure portrayed in a sequential access.

Fig.4



To fully realize the interaction between the user and the system we will make use of several **windows navigation diagrams** to help visualize the basic components of the interface, their states and how the communicate with each other not forgetting the corresponding transition paths. Shown below is a transaction centered windows navigation diagram.

F ig.5



It is **important** to note that both the wire frame and the windows navigation diagrams shown above are only preliminary designs and they depict only **a predicted s**ystem structure and navigation which is **open to change** during the course of the actual coding.

2.2. Screen Standards

Human Computer Interaction layer design is an art whose overall goal is to produce a simple, eye catching interface that maximises efficiency of use of the system in question. Discussed below are a few **standards / principles** complimented by **Ben Shneiderman's 8 golden rules** of interactive interface design which we followed in designing components involved in navigation, input, and output which make up the Interface. These standards are important in balancing a simple, pleasant appearance and in turn maximizing the efficiency with the intended user circumnavigates and uses the system.

Layout and Aesthetics

The interface will be designed as a series of areas on the screen. Separate areas of the interface will be designated for information output, input and navigation. The interface should be functional with a pleasant appearance which means a careful use of colours, white spaces and fonts. Generally we will try to avoid busy interfaces by **ensuring visibility of system states** thereby **minimising user memory load.**

Content Awareness

In designing the system interface we will ensure that users know exactly where they are in the system. To ensure all information displayed is clear **simple error handling mechanisms** and **error messages** will be implemented and all information will be worded in **the user's language.**

User Experience and Minimal User Effort

The user's efficiency greatly depends on the ease of use of the interface. To realize a simple to use interface we will make use of shortcuts and limit the number of clicks. **Provision of shortcuts** will speed up the interaction for the experienced user allowing the system to cater for both the experienced user and the beginner with shortcuts and a simple interface respectively. Ease of use and learning will allow for a greater **user experience** preventing boredom and fatigue from such a repetitive process like hotel reservations. This will be done by the **provision of clearly marked exits and easy action reversal.**

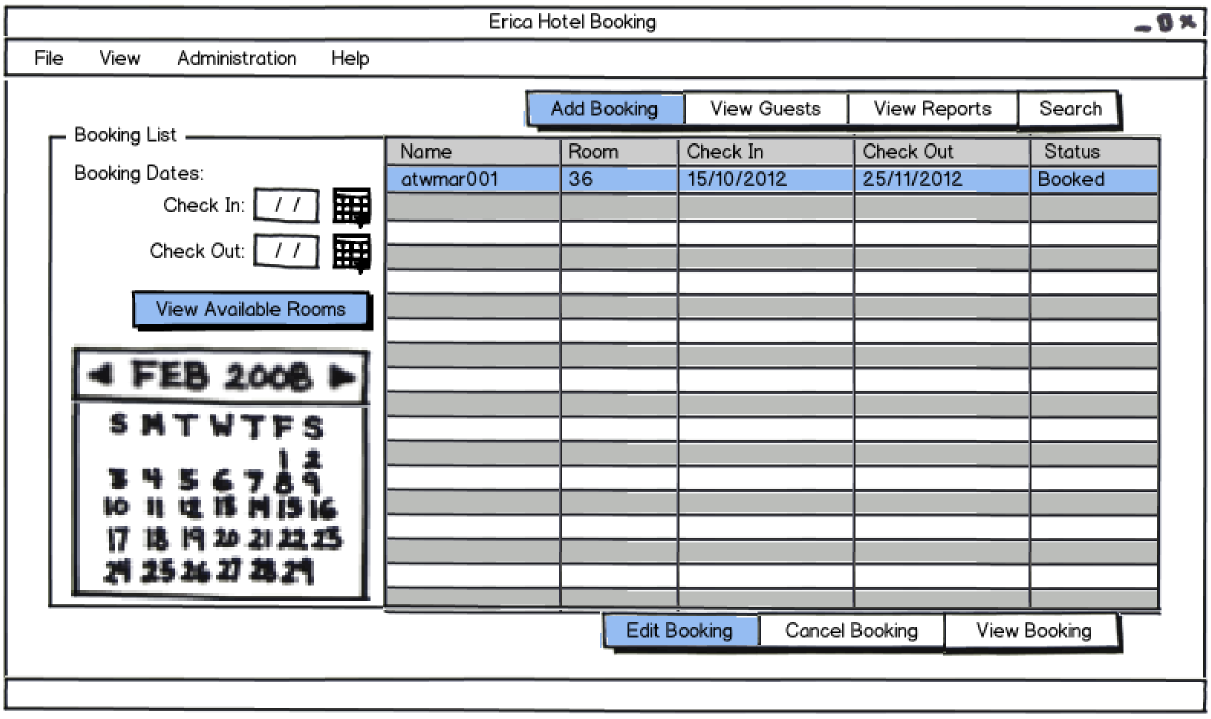
Consistency

The interface components such as navigation buttons will be designed with uniformity allowing for consistency in the interface design giving users a good idea of what happens when they perform a particular function. **Being Consistent** prevents any uncertainties whether different text, buttons or actions represent the same thing.

2.3. Detailed Screen Layout

In designing detailed screen layouts, the emphasis will be placed on how the user interacts with the system through three major parts namely navigation components such as buttons, menus, input components such as forms, text boxes and output components such as reports. It is important here to note that all screens detailed below are not in any particular order of access and are only realistic mockups open to change during system coding. Input and Output standards and controls of components on the screens are discussed in a later section.

F ig.6



Booking List

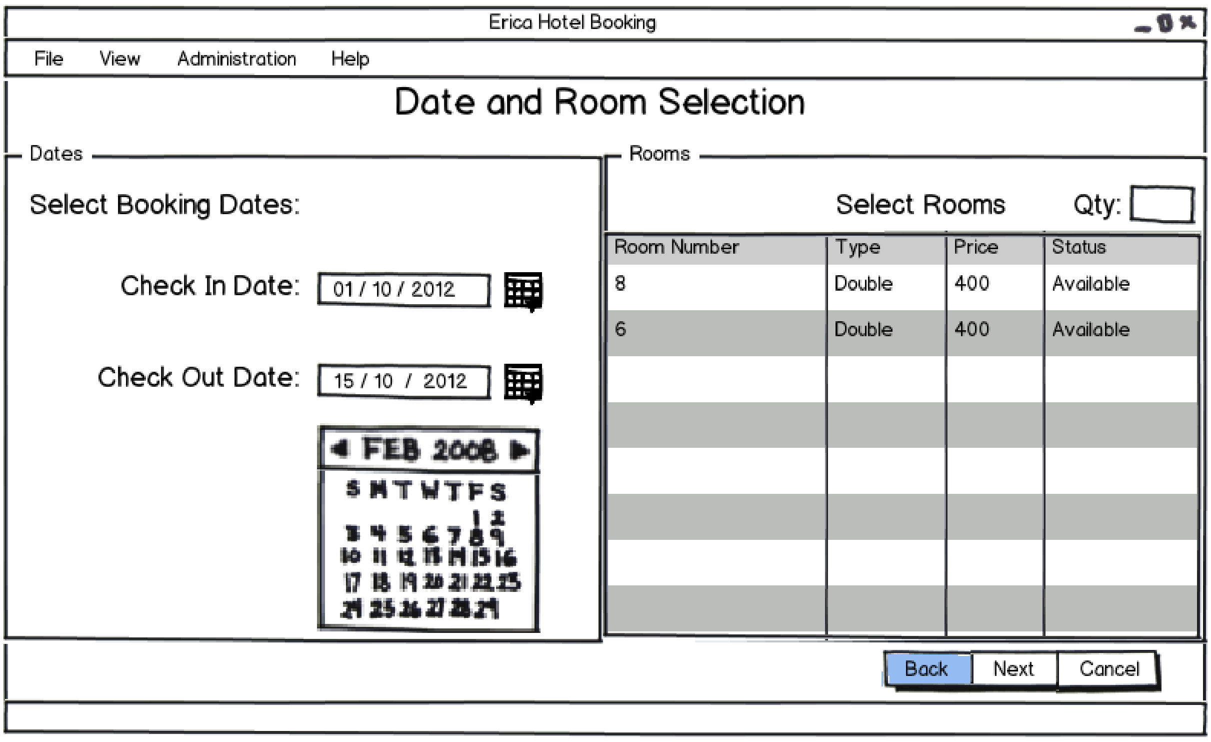
Booking Dates:

Check In: / /

Check Out: / /

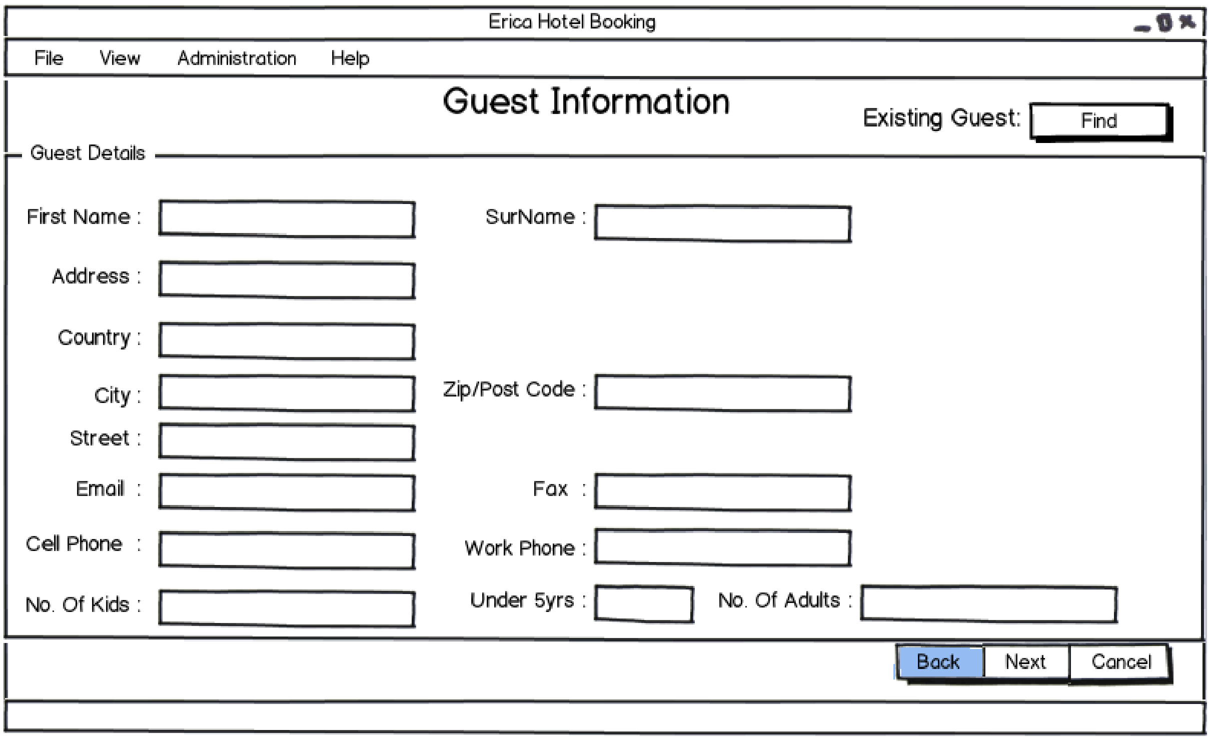
Name	Room	Check In	Check Out	Status
atwmar001	36	15/10/2012	25/11/2012	Booked

F ig.7

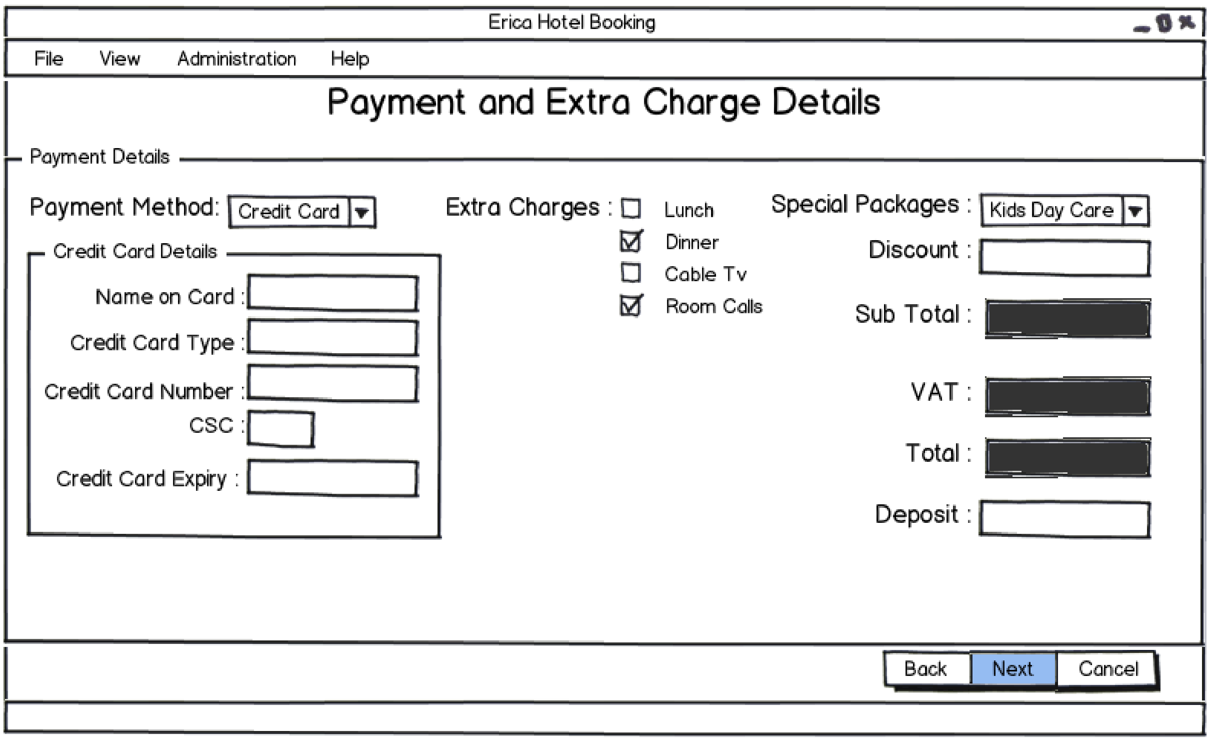


Room Number	Type	Price	Status
8	Double	400	Available
6	Double	400	Available

F ig.8



F ig.9



Erica Hotel Booking

File View Administration Help

Payment and Extra Charge Details

Payment Details

Payment Method: Credit Card ▼

Credit Card Details

Name on Card :

Credit Card Type :

Credit Card Number :

CSC :

Credit Card Expiry :

Extra Charges : ☐ Lunch ☑ Dinner ☐ Cable Tv ☑ Room Calls

Special Packages : Kids Day Care ▼

Discount :

Sub Total :

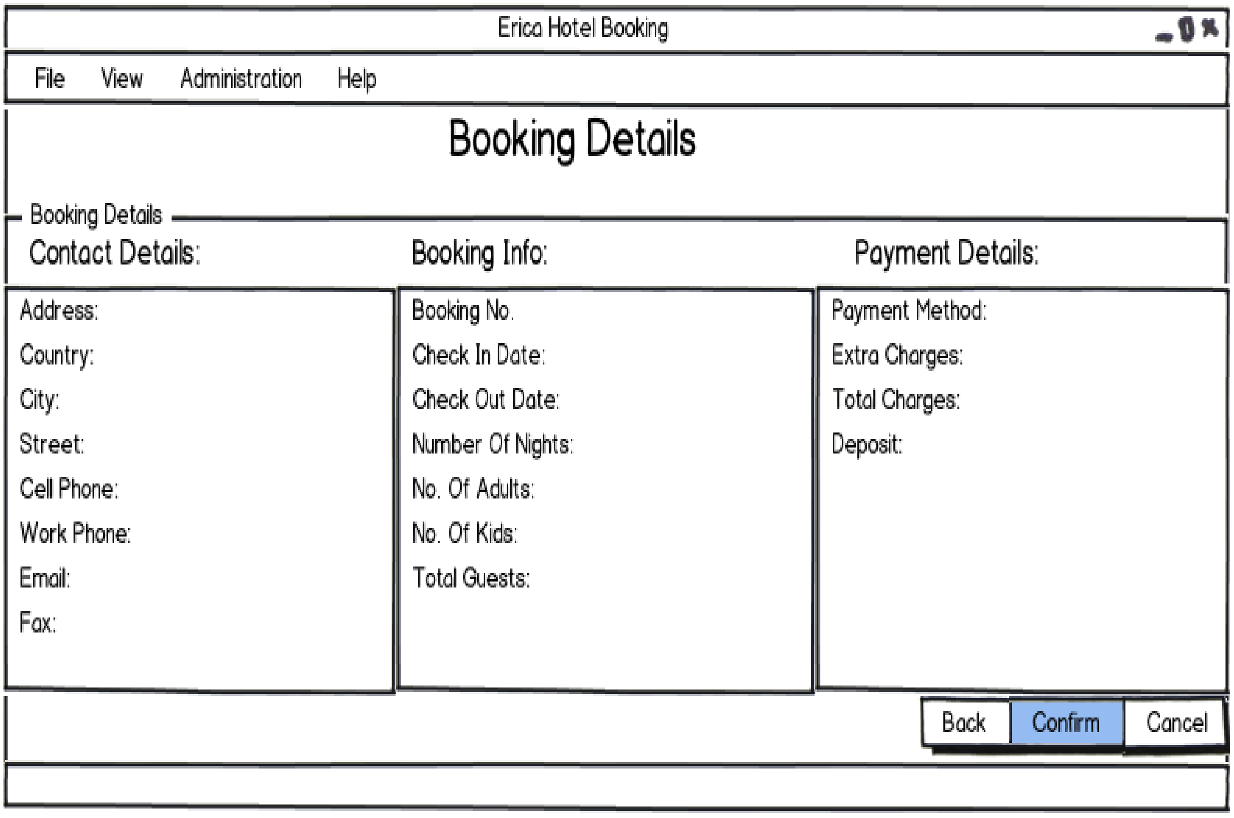
VAT :

Total :

Deposit :

Back Next Cancel

F ig.10



Erica Hotel Booking

File View Administration Help

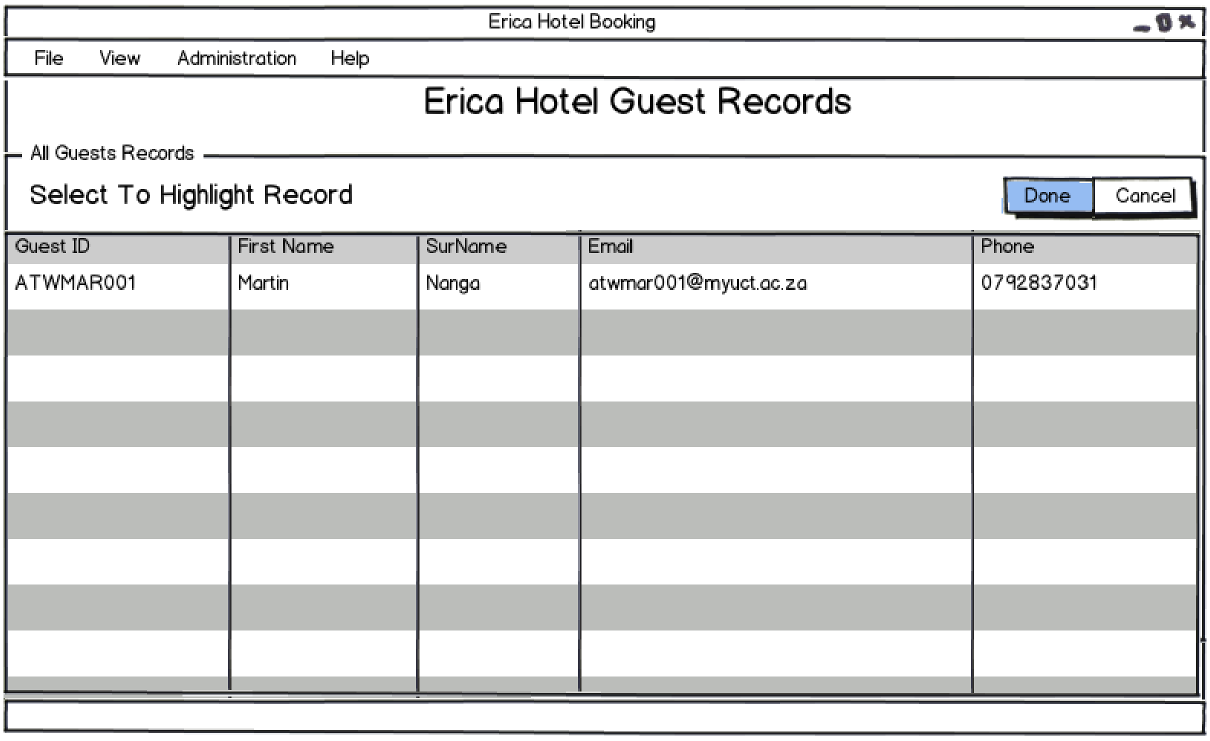
Booking Details

Booking Details

Contact Details:	Booking Info:	Payment Details:
Address:	Booking No.	Payment Method:
Country:	Check In Date:	Extra Charges:
City:	Check Out Date:	Total Charges:
Street:	Number Of Nights:	Deposit:
Cell Phone:	No. Of Adults:	
Work Phone:	No. Of Kids:	
Email:	Total Guests:	
Fax:		

Back Confirm Cancel

F ig11



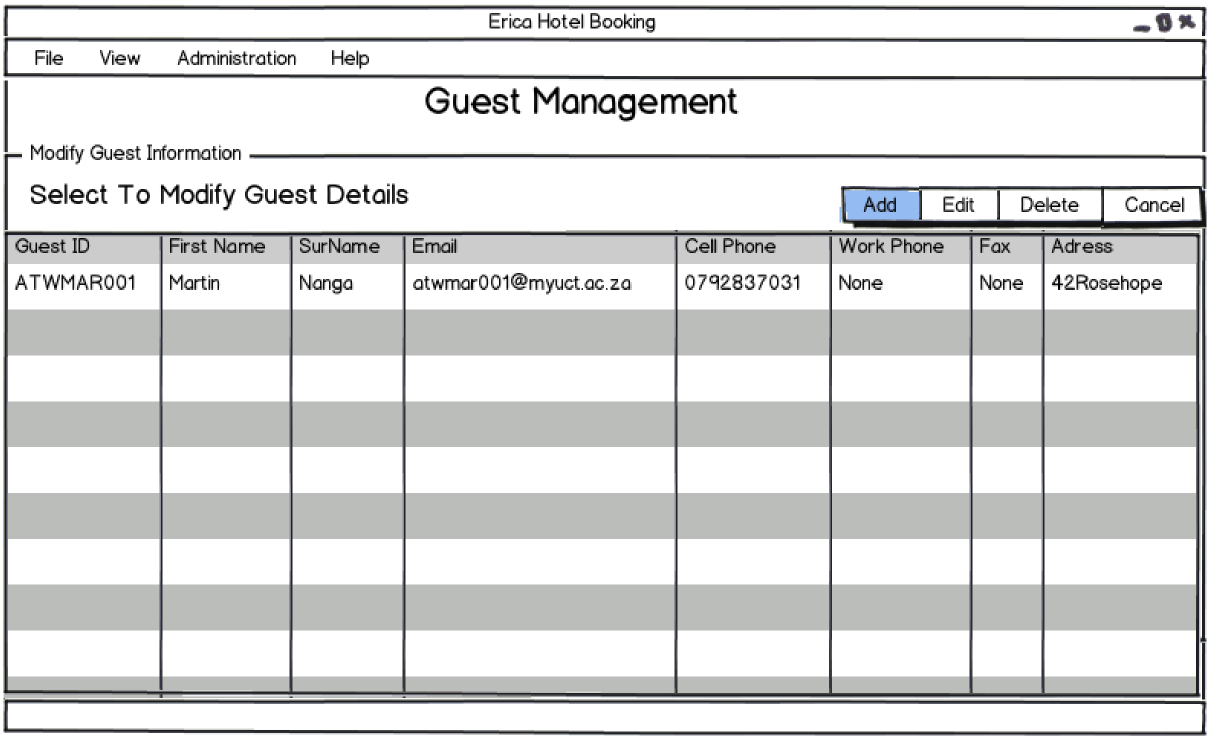
Erica Hotel Guest Records

All Guests Records

Select To Highlight Record

Guest ID	First Name	SurName	Email	Phone
ATWMAR001	Martin	Nanga	atwmar001@myuct.ac.za	0792837031

F ig.12



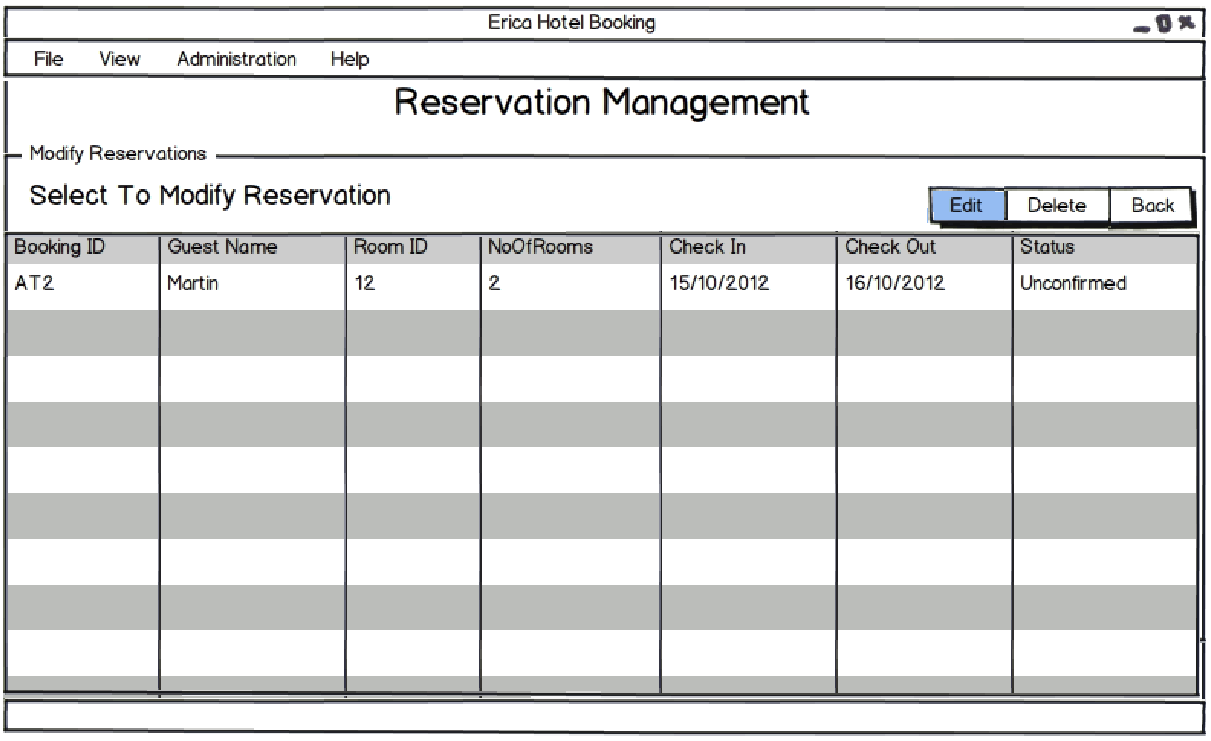
Guest Management

Modify Guest Information

Select To Modify Guest Details

Guest ID	First Name	SurName	Email	Cell Phone	Work Phone	Fax	Adress
ATWMAR001	Martin	Nanga	atwmar001@myuct.ac.za	0792837031	None	None	42Rosehope

F ig.13



Booking ID	Guest Name	Room ID	NoOfRooms	Check In	Check Out	Status
AT2	Martin	12	2	15/10/2012	16/10/2012	Unconfirmed

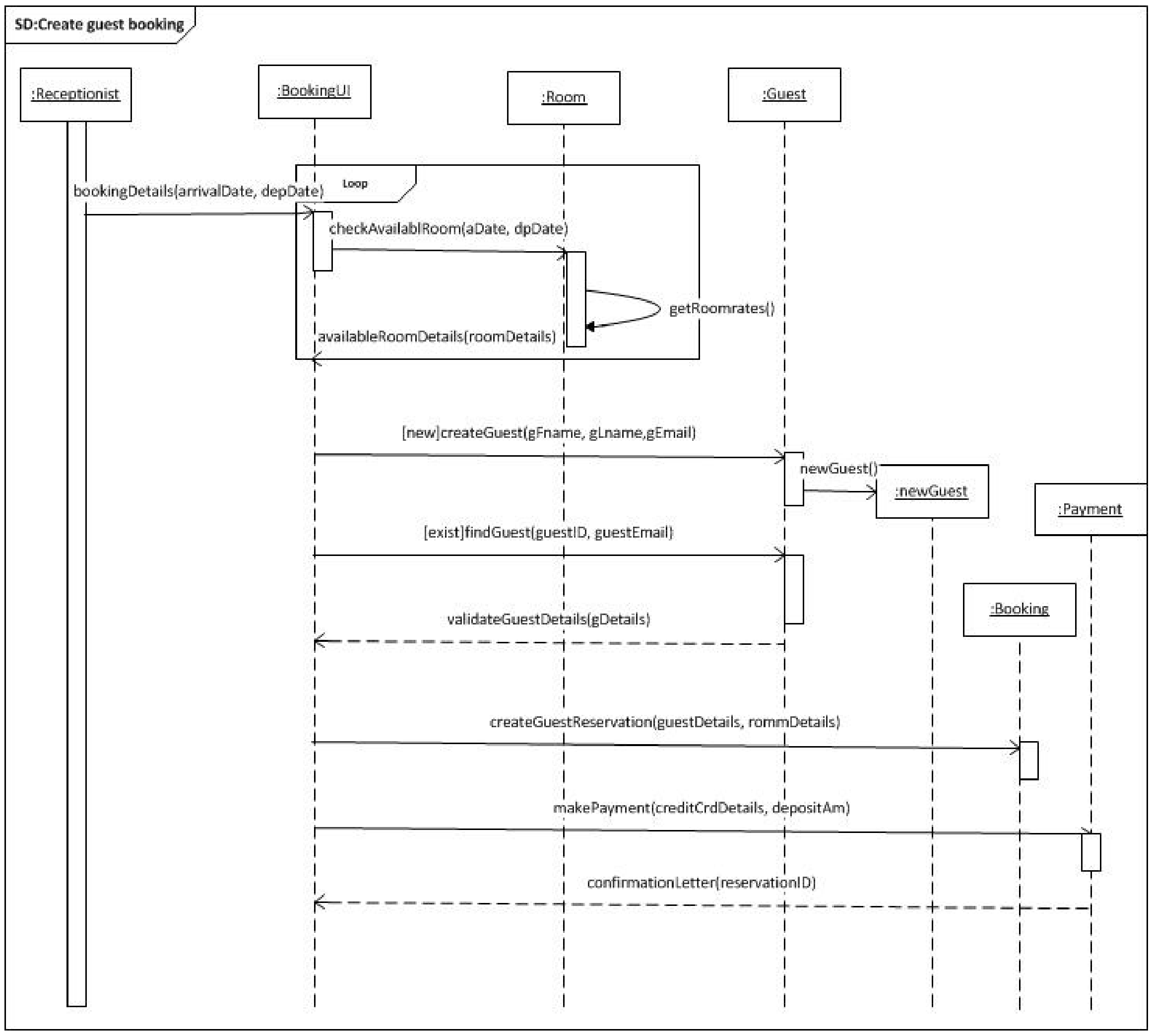
Mock Up Layout Dictionary:

Figure	Usage
6	This is the Main Menu Form from which all navigation originates also displays reservation records
7	Capture room choices and booking dates
8	Captures Guest details during booing process
9	Captures Payment Deposit and displays subtotals
10	Displays all captured data before making a booking
11	View All Erica Hotel Guests
12	Edit, Delete, Add New Guests using this form
13	Edit, Delete Reservations using this Form

3. Design Sequence Diagrams

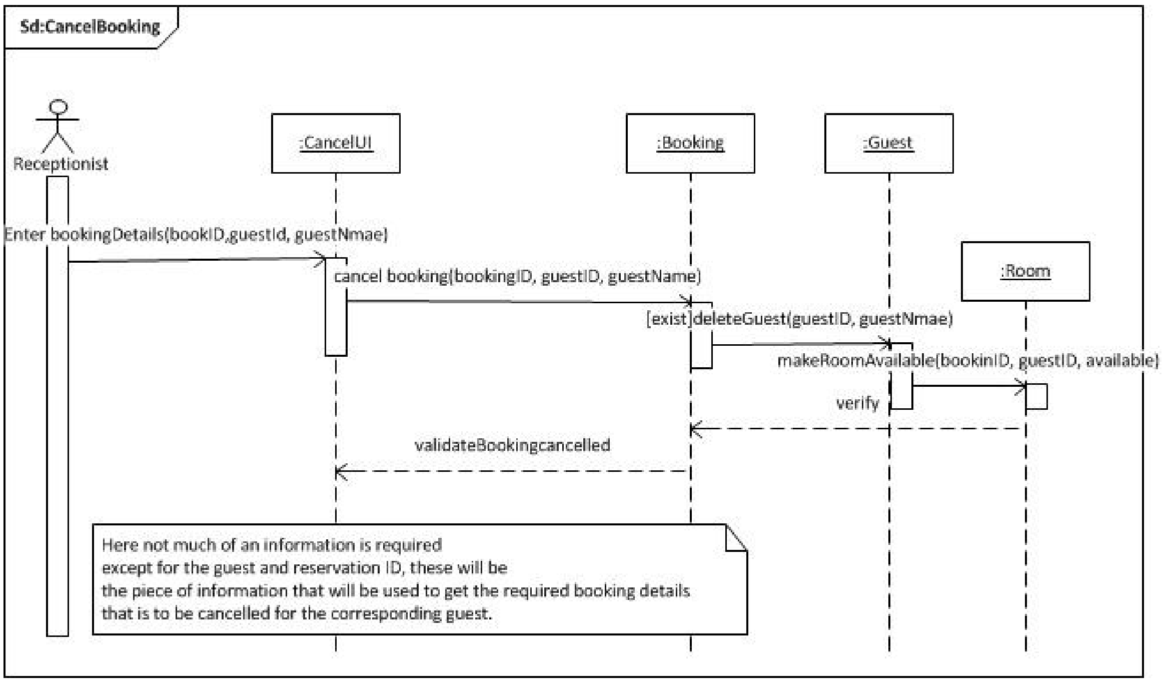
The customer provides the receptionist with the dates of preference of stay and these are then entered into the system to check the availability of the rooms for the specified dates. Provided that the rooms are available the customer is informed and a new guest is created for the room. Then the receptionist confirms guest details and creates the booking for the guest.

3.1. Make a booking Design Sequence Diagram

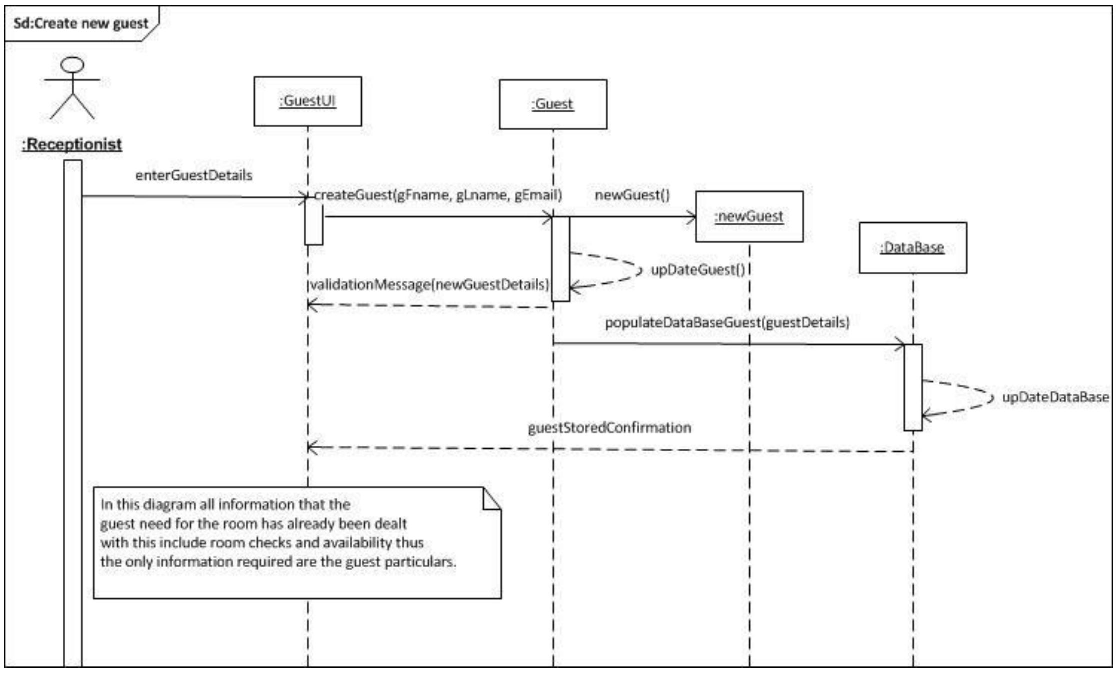

SD:Create guest booking
:Receptionist
:BookingUI
:Room
:Guest
Loop
bookingDetails(arrivalDate, depDate)
checkAvailablRoom(aDate, dpDate)
getRoomrates()
availableRoomDetails(roomDetails)
[new]createGuest(gFname, gLname,gEmail)
newGuest()
:newGuest
:Payment
[exist]findGuest(guestID, guestEmail)
:Booking
validateGuestDetails(gDetails)
createGuestReservation(guestDetails, rommDetails)
makePayment(creditCrdDetails, depositAm)
confirmationLetter(reservationID)

3.2. Cancel booking Design Sequence Diagram

In this diagram the booking ID is mainly used to get other piece information linked to the reservation such as guest and room ID. These are then used on instance level after the receptionist has pressed the cancel button to cancel the reservation and delete data in guest and room instances but keep some of the guest particulars for future returns of the guest.

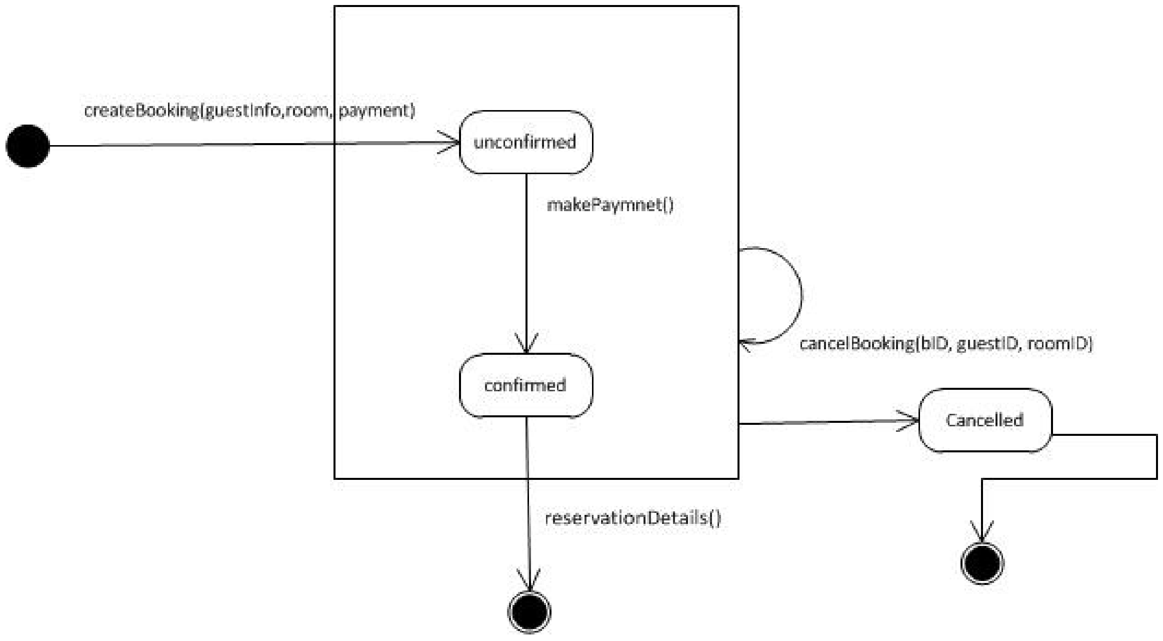

Sd:CancelBooking
Receptionist
:CancelUI
:Booking
:Guest
:Room
Enter bookingDetails(bookID,guestId, guestNmae)
cancel booking(bookingID, guestID, guestName)
[exist]deleteGuest(guestID, guestNmae)
makeRoomAvailable(bookinID, guestID, available)
verify
validateBookingcancelled
Here not much of an information is required except for the guest and reservation ID, these will be the piece of information that will be used to get the required booking details that is to be cancelled for the corresponding guest.

3.3. CREATE GUEST DESIGN SEQUENCE DIAGRAM

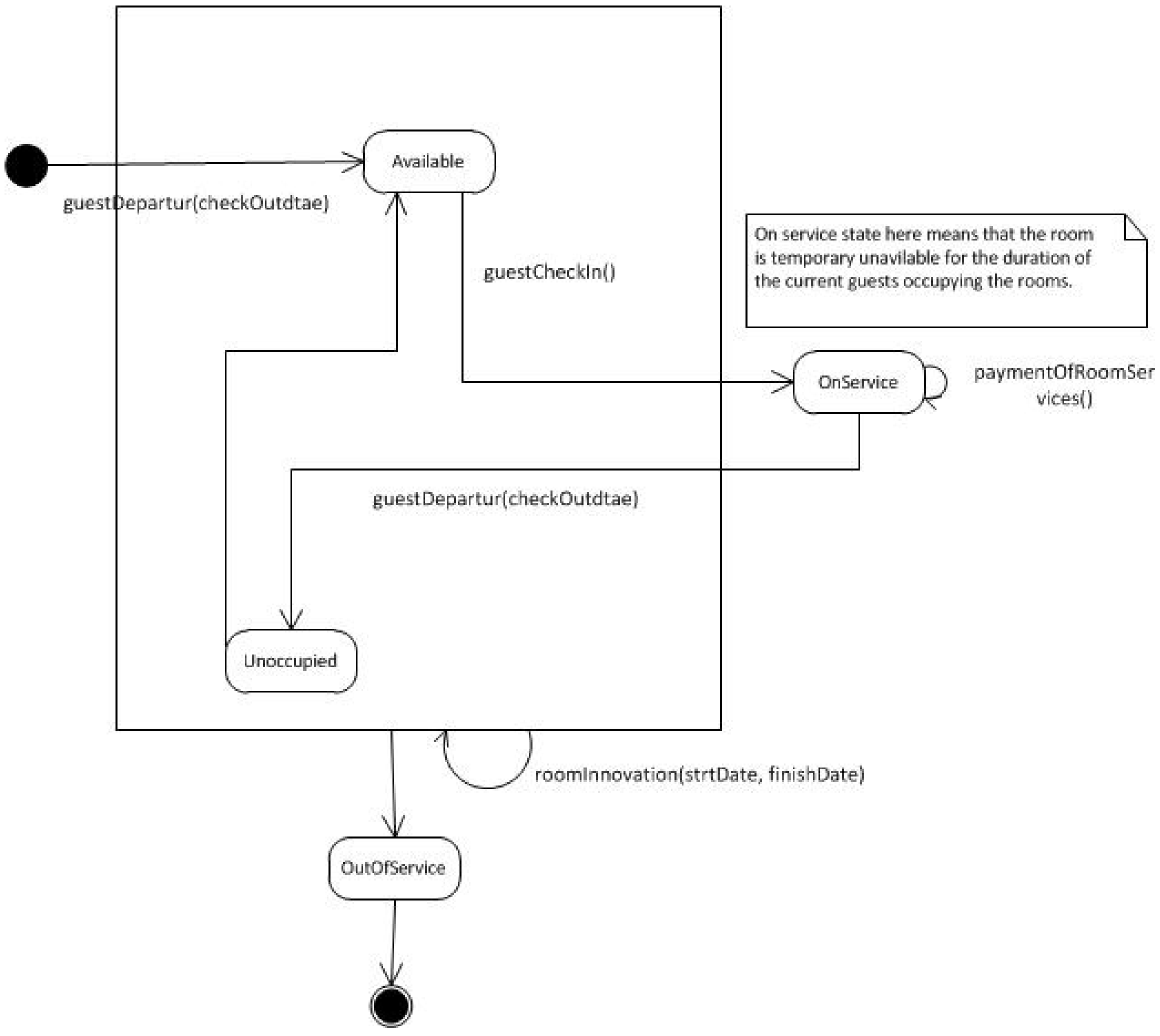

Sd:Create new guest
:Receptionist
:GuestUI
:Guest
:newGuest
:DataBase
enterGuestDetails
createGuest(gFname, gLname, gEmail)
newGuest()
upDateGuest()
validationMessage(newGuestDetails)
populateDataBaseGuest(guestDetails)
upDateDataBase
guestStoredConfirmation
In this diagram all information that the guest need for the room has already been dealt with this include room checks and availability thus the only information required are the guest particulars.

4. DESIGN STATE MACHINE DIAGRAMS

4.1. BOOKING INSTANCE STATE MACHINE DIAGRAM


createBooking(guestInfo,room, payment)
unconfirmed
makePaymnet()
confirmed
cancelBooking(bID, guestID, roomID)
Cancelled
reservationDetails()

4.2. Room Instance State Machine Diagram


Available
guestDepartur(checkOutdtae)
guestCheckIn()
On service state here means that the room is temporary unavilable for the duration of the current guests occupying the rooms.
OnService
paymentOfRoomServices()
guestDepartur(checkOutdtae)
Unoccupied
roomInnovation(strtDate, finishDate)
OutOfService

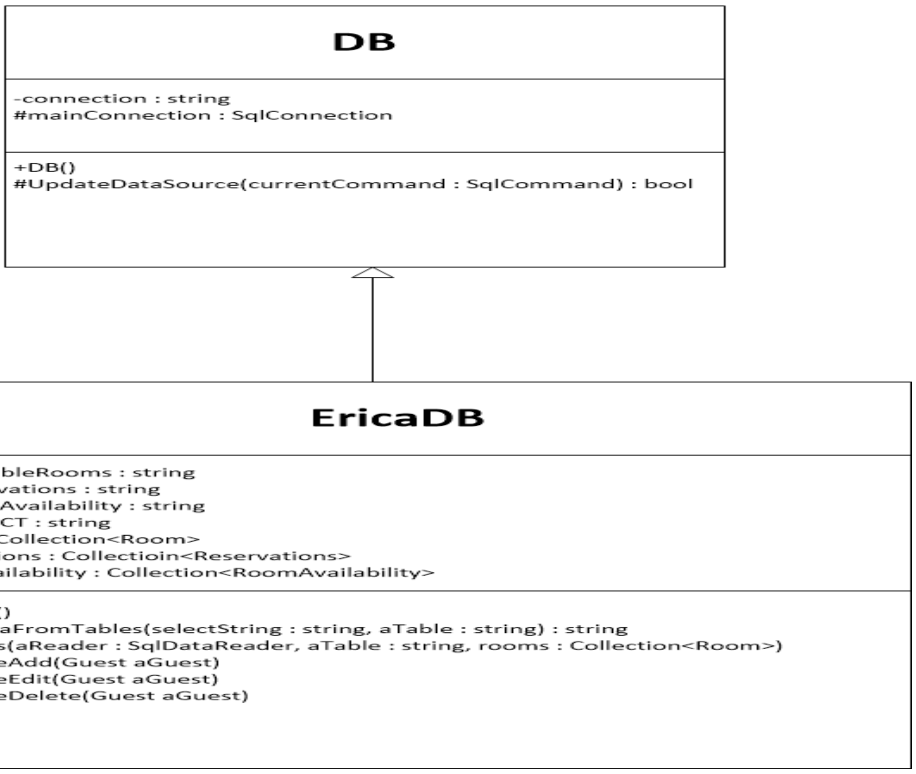
5. Design Class Diagrams

The design class diagrams illustrated in this section are a completion of the analysis versions of the same diagrams with a lot more detail. In designing our class diagrams our overall aim is to show the design components such as attributes, objects and methods within the classes, their visibility and relationships to other classes.

Given that were making use of a three – tier architecture approach as portrayed in the package diagram seen earlier, we will model the classes in each layer separately at first and later explain how the work hand-in hand to achieve a particular function.

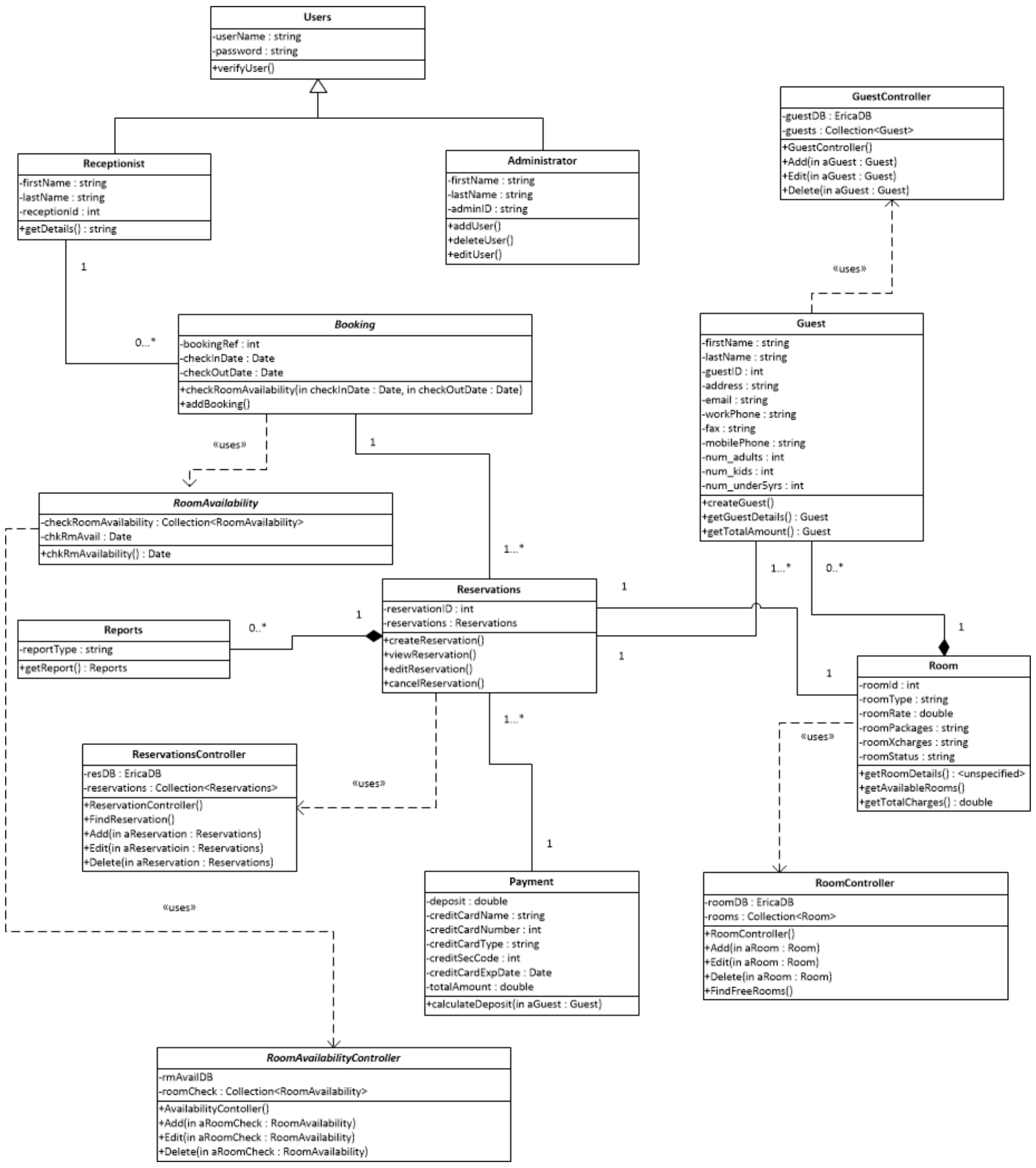
5.1. Data Layer Design Class Diagrams

This Layer contains a set of data access classes and is used to read and write data to a given data source. The methods in these classes encapsulate logic needed to obtain data from the business logic classes and copy it to the data source and vice versa. The **EricaDB** class which is responsible for reading data from the tables in the database inherits the **DB** class which is responsible for creating a connection to the database.


DB
-connection : string
#mainConnection : SqlConnection
+DB()
#UpdateDataSource(currentCommand : SqlCommand) : bool
EricaDB
-tblAvailableRooms : string
-tblReservations : string
-tblRoomAvailability : string
-sql_SELECT : string
-rooms : Collection<Room>
-reservations : Collectioin<Reservations>
-roomAvailability : Collection<RoomAvailability>
+EricaDB()
-ReadDataFromTables(selectString : string, aTable : string) : string
-FillTables(aReader : SqlDataReader, aTable : string, rooms : Collection<Room>)
-DatabaseAdd(Guest aGuest)
-DatabaseEdit(Guest aGuest)
-DatabaseDelete(Guest aGuest)

5.2. BUSINESS LOGIC LAYER DESIGN CLASS DIAGRAMS

This Layer contains a set of **domain logic classes** which are used to map data from the data source to an in memory object in this case a collection object and **controller classes** which are used to manage the view/display objects associated with the user interface.


Users
-userName : string
-password : string
+verifyUser()
Receptionist
-firstName : string
-lastName : string
-receptionId : int
+getDetails() : string
Administrator
-firstName : string
-lastName : string
-adminID : string
+addUser()
+deleteUser()
+editUser()
GuestController
-guestDB : EricaDB
-guests : Collection<Guest>
+GuestController()
+Add(in aGuest : Guest)
+Edit(in aGuest : Guest)
+Delete(in aGuest : Guest)
«uses»
1
0...*
Booking
-bookingRef : int
-checkInDate : Date
-checkOutDate : Date
+checkRoomAvailability(in checkInDate : Date, in checkOutDate : Date)
+addBooking()
Guest
-firstName : string
-lastName : string
-guestID : int
-address : string
-email : string
-workPhone : string
-fax : string
-mobilePhone : string
-num_adults : int
-num_kids : int
-num_under5yrs : int
+createGuest()
+getGuestDetails() : Guest
+getTotalAmount() : Guest
«uses»
1
RoomAvailability
-checkRoomAvailability : Collection<RoomAvailability>
-chkRmAvail : Date
+chkRmAvailability() : Date
1...*
1...*
0..*
1
Reservations
-reservationID : int
-reservations : Reservations
+createReservation()
+viewReservation()
+editReservation()
+cancelReservation()
Reports
-reportType : string
+getReport() : Reports
0..*
1
1
1
1
Room
-roomId : int
-roomType : string
-roomRate : double
-roomPackages : string
-roomXcharges : string
-roomStatus : string
+getRoomDetails() : <unspecified>
+getAvailableRooms()
+getTotalCharges() : double
«uses»
1...*
ReservationsController
-resDB : EricaDB
-reservations : Collection<Reservations>
+ReservationController()
+FindReservation()
+Add(in aReservation : Reservations)
+Edit(in aReservatioin : Reservations)
+Delete(in aReservation : Reservations)
«uses»
1
Payment
-deposit : double
-creditCardName : string
-creditCardNumber : int
-creditCardType : string
-creditSecCode : int
-creditCardExpDate : Date
-totalAmount : double
+calculateDeposit(in aGuest : Guest)
RoomController
-roomDB : EricaDB
-rooms : Collection<Room>
+RoomController()
+Add(in aRoom : Room)
+Edit(in aRoom : Room)
+Delete(in aRoom : Room)
+FindFreeRooms()
«uses»
RoomAvailabilityController
-rmAvailDB
-roomCheck : Collection<RoomAvailability>
+AvailabilityContoller()
+Add(in aRoomCheck : RoomAvailability)
+Edit(in aRoomCheck : RoomAvailability)
+Delete(in aRoomCheck : RoomAvailability)

5.3 Presentational layer Design Class Diagrams

The diagram below illustrates the interface classes and their various relationships and dependencies.

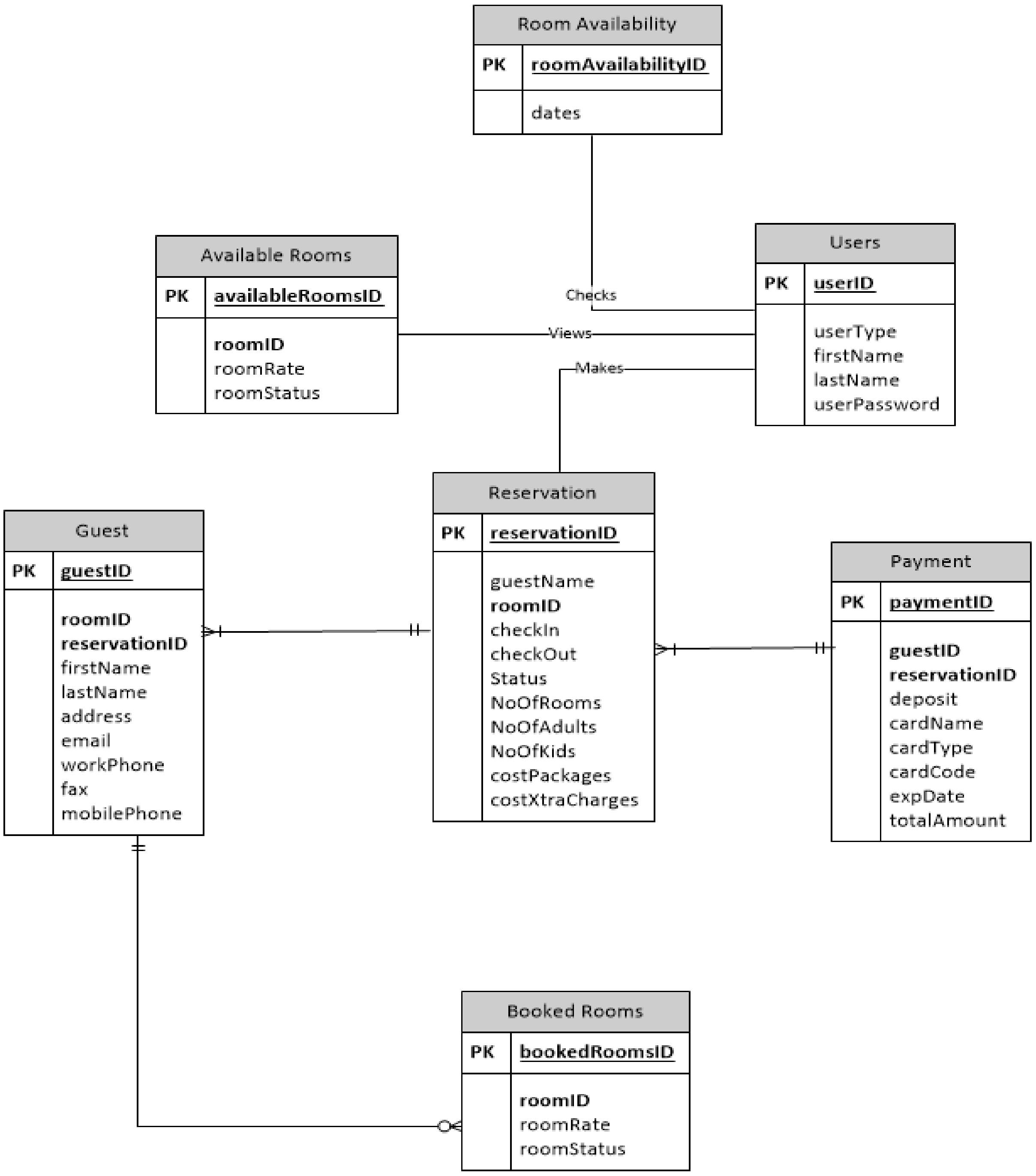
5.4 FULL SYSTEM DESIGN CLASS DIAGRAMS.

The diagram below is a combination of all the classes in the different layers full showing all dependencies and relationships.

In summary, the **user interface classes** request or submit data to **the controller classes** which in turn request or submit data to **the domain classes** which are responsible for mapping data into an in memory object by requesting the data layer **DB classes** to retrieve or submit data to the data source.

6. Entity Relationship Diagram

The system database will be implemented using a normalized relational database. Classes with simple data structure will become tables and object Ids will become primary keys. Where classes contain another class as an attribute, a separate table will be created for the embedded class. For Collections we will create two tables, one to hold object Ids and the other to hold the objects in the collection. The tables will be normalized to Third Normal Form where every attribute is dependent on the primary key and not on another non – key attribute. Below is a 3NF Entity Relationship Diagram show all cardinalities and connectivity providing a detailed view of all attributes, keys, data types and field sizes.



7. Report Design

7.1. Report 1

This report has the information about the number of occupied and not occupied rooms for a given specified dates. Information regarding an amount of money that the prospective guest would be required to pay for a certain duration of stay that the customer plans on spending on the hotel.

For the user of the system it is more valuable to provide an information that is ready to be used such as the one that will be on this report. The user will not have to do calculations for a certain amount of money that will be for a certain period of stay requested by the customer and also will not have difficulty on providing this information to the prospective guest, and consequently this will save the time for the user from doing literal calculations regarding room rates and stay thus speeding up the booking process.

This report will be on customer's request based on the specified dates of stay. Thus it will be the receptionist that will be on disposal of this report quite oftenly.

7.1.1. Detailed Output Requirements

- **Output type & ID:** Detailed occupancy report.
- **Report Objectives:** The main objective is to provide comprehensive information about occupancy and room rates that the user can use to answer such related questions of the customers.
- **Audience:** Receptionist and the accounting manager.
- **Content:** Columns: RoomID, Occupancy status, prospective guest, total cost of stay, nights and dates.
- **Layout:** Columnar report.
- **Selection:** All occupied rooms for a certain requested period grouped together differently from non-occupied rooms.
- **Sequence:** Descending by length of stay.
- **Comparison:** Long periods of stay >= Short periods of stay.
- **Grouping/Summarisation:** Information regarding the occupied rooms and their rates with total cost of stay will be grouped under the occupied category otherwise not occupied category.
- **Media to be used:** Electronic media display.
- **Frequency, Timing, Delivery:** On daily basis and on prospective guest request.

- **Distribution:** The receptionist and probably an accounting manager for financial calculations..
- **Private, security & integrity requirements:** None of these requirements will be required.

- ***Report Layout: Columns***

<u>Occupancy room rates and total cost of stay report.</u>

-

• <u>AVAILABLE ROOMS REPORT</u>

In this report all the information regarding the available rooms, their corresponding rates and number is presented.

The valuability of this report is that , it makes it easy for the receptionist to provide the prospective guest with the information about rooms that are available and their corresponding rates and room types. Receptionist can also provide alternative stay dates for the customer based on what has been returned by the report whether for the required dates the rooms are available or not..

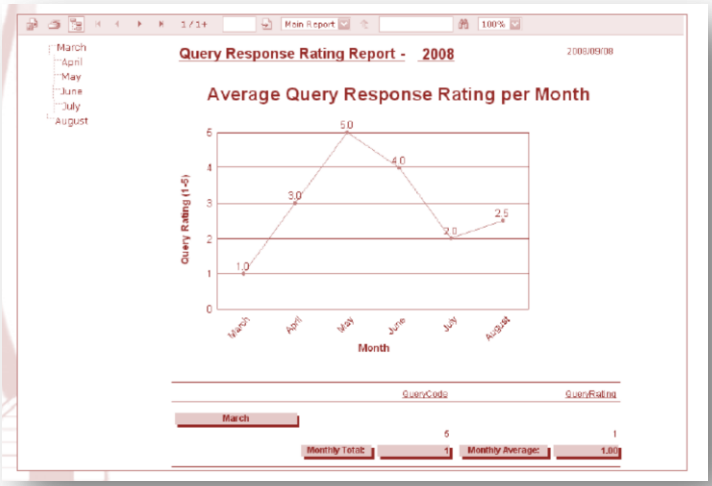
The receptionist will generate this report using the dates provided by the prospective guest to find the rooms that are on offer for that period. These available rooms will be displayed on screen together with their corresponding rates.

- ***Detailed Output requirements***
  - **Output type & ID:** Available rooms summery
  - **Report Objectives:** Presenting all of the available rooms so that quick responses to customers and identification of non-availble rooms can be made. Alternative options to try other dates in which rooms will be available can be suggested to the customer.
  - **Audience:** Receptionist
  - **Content:** Column presented: room number, room type, room rate, room brief description, availability status.

- **Layout:** Columnar report.
- **Selection:** As all these are available rooms, the selection will be based on room types. Rooms with high rates will follow the ones with low rates.
- **Sequence:** Ascending in rates.
- **Comparison:** The low rate rooms <= The high rate rooms.
- **Grouping/Summarisation:** Grouped according to rates and their room numbers.
- **Media to be used:** Electronic medium.
- **Frequency, Timing, delivery:** On prospective guest request and every beginning of the month.
- **Distribution:** The receptionist and the hotel manger.
- **Privacy, security & integrity requirements:** None.

- ***Report Layout: Column***

7.1.2. Report Layout



Query Response Rating Report - 2008 2008/09/08

	QueryCode		QueryRating
March			
	5		1
Monthly Total:	1	Monthly Average:	1.00

8. Input-Output Standards & Controls

To protect the new system data, it is important to implement integrity and security controls. Designing of input and output controls focuses the content to be displayed for or collected from the user by the system user interface. In designing input – output controls our overall aim is to put in place feedback mechanisms that return a system output to a user input informing the user whole the system is reacting to the particular input or request.

8.1. Formalised Outputs:

All major outputs and reports will be displayed on screen. The designed system will make use of a **list view** control to display a summary of the major output reports such as available rooms, guests and reservations. Similarly to confirmation reports, hotel occupancy reports will also be displayed on screen with an additional print out functionality.

8.2. Standards for User Interfaces

User interface standards will specify how input, output and navigation mechanisms work. For navigation we made use of on form **command buttons** and **menu strips**. Backwards and forward navigation buttons were also implemented. To uniformly arrange elements on the form we made use of group boxes to group similar content in a specific area of the screen.

8.3. Standards for Dialogues

Dialogue design is important to inform the user of the state of the system during operation. We have implemented standards to yield meaningful messages in case of errors, system malfunction or delayed processing.

8.4. Standards for Controls

Implementing standards for controls ensures that erroneous data is screen out before it enters the system. All input fields have validation mechanisms to reject any invalid entries.

Output controls to ensure out reports are displayed at the right time and place have also been implemented.

8.5. BUILT-IN VALIDATION TO ENSURE REQUIREMENTS ARE MET

Audit controls such feedback messages on a successful action have been implemented. It is important to note that these controls will not be looking for errors but will check to see if a particular user action has been successful. One of the audit controls implemented is a feedback message in response to a database modification procedure to notify the user on a success or failure scenario. A list view to monitor data changes in the database has also been implemented allowing for real time verification of any action performed on the data.

8.6. INPUT INTEGRITY CONTROLS

These controls ensure that quality data is entered into the system. We implemented input integrity controls by making use of methods to reduce common errors. We have implemented a date picker control, a drop down list and radio buttons to limiting the need to always type in all data.

8.7. OUTPUT INTEGRITY CONTROLS

To ensure quality output we have implemented list views and grid views to fully format data before display. Verification messages have also been implemented to confirm successful display of information. All output reports will be on screen read only to prevent unwanted access.

9. IMPLEMENTATION PLAN

The main purpose of this implementation plan is to ensure the new system deployment and transition occurs flawlessly and efficiently. The implementation phase tasks us with making the new system available to the given users as well as putting into place appropriate measures for ongoing support allowing for short term system maintenance. The new system will be **deployed** using a phased implantation in a pilot mode. Here we will roll out the system to a small group of users in the Erica Hotel Group. Once one Hotel in the group successfully transitions from the old system to the new system we can roll the system to another group of users at another hotel in the group. This type of implementation will allow for more testing to be done before rolling the system to the whole group. The only disadvantage is that the deployment process will be stretched out prolonging the implementation period. The process of **deploying the new system** will be accomplished in **three steps**. The first step is to **prepare system for implementation.** This will involve ensuring all intended users have the required system access and security passes, ensuring all components interfacing with the new system are in the same environment and distribution of materials such as training manuals. The second step is **deploying** the new system and this will involve successfully preparing the users to assume responsibility for system maintenance and support as long time support is out of the question, user training sessions and parallel running of the new and old system within the chosen small group. Last but not least the **Transition step,** this will involve moving from developer to offering short term system support in essence putting an end to the design phase.

10. TEST PLAN

The overall goal in testing is to ensure the system performs as designed and more importantly that it achieves all the functional requirements. By investing all system components testing allows for the location and correction of errors. In this section we are going to establish testing starndards, describe and estimate the testing process and also establish the needed software and hadware.

10.1. TEST ENVIRONMENT

To run the new system for testing the computer system in to be used should meet the following requirements needed to install **Microsoft Visual studio 2010.**

Hardware Requirements

- Computer that has a 1.6GHz or faster processor
- 1 GB **(32 Bit)** or 2 GB **(64 Bit)** RAM (Add 512 MB if running in a virtual machine)
- 3GB of available hard disk space
- 5400 RPM hard disk drive
- DirectX 9 capable video card running at 1024 x 768 or higher-resolution display

- DVD-ROM Drive

Software Requirements

- Windows XP (x86) with Service Pack 3 - all editions except Starter Edition
- Windows Vista (x86 & x64) with Service Pack 2 - all editions except Starter Edition
- Windows 7 (x86 & x64)
- Windows Server 2003 (x86 & x64) with Service Pack 2
- Windows Server 2003 R2 (x86 & x64)
- Windows Server 2008 (x86 & x64) with Service Pack 2
- Windows Server 2008 R2 (x64)
- Windows 8 if you can afford a genuine copy…just kidding!

10.2. TEST ITEMS

Provide a description of all the features to be tested

10.3. TEST APPROACHES

Testing will be performed using a **gray-box** approach which is a combination of both white and black box testing approaches. This testing approach focusses on functional testing and system code, flow and logical structure testing. A top down strategy which starts with the general controls at **the top level through to the bottom level** will also be employed. Below is a description of how the various testing levels relate to each other.

Unit Testing

This involves testing a single module or component. We use test data to check the behavior of the chosen unit irrespective of its relationship with other units. This type of testing will be done parallel to developing the system as each new added unit will be thorough tested.

Function and System Testing

This will be performed on a partially finished system using simulated data to test for each functional requirement. For example the add a booking function will be tested as soon as it is completed and will be followed by other functions such as cancel booking and so on. The

system testing will then be performed on a completely finished system using the test items seen earlier.

User acceptance Testing

This type of testing will be performed on the entire complete system after it is set-up at the client's site. This will make use of real data provided by the client and will encompass an **alpha**, **beta** and **gamma** test. The alpha test will be in a controlled environment where we will exhibit major system functions. The beta test will allow for real users unfamiliar with the system to try out a pre-release beta version of the system. The gamma test will test for system compatibility checking how the new system integrates with the old existing system.

Systems performance Testing

This testing will greatly focus on the behavior of the new system. This will involve testing for non-functional requirements such as system efficiency and how well it process large amounts of data during peak seasons. Test cases here involve emergency situations such as database failure.

Audit Testing

This will be performed last and its purpose is to determine whether the system is free of errors. This will have to be performed by a quality assurance agent who will employ a white box approach to test all system components.

10.4. PROBLEM TRACKING & RESOLUTION

To ensure quality of the system, a record of all errors was kept. Common errors were identified and their resolutions note. This was helpful when we encountered similar errors further on in the development process as we didn't have to waste time coming up with new fixes. A record of all the actions that resulted into errors was kept and used for the final system testing. The error records kept included **status values** to describe errors such as **new** and **resolved**. Problems were resolved by addressing their root causes through the identification viable solutions. We also made use of build numbers to identify earlier versions of the system as well as in code comments and IDE bug tracking mechanisms to incorporate breaks and data value run through.

10.5. Test Schedule

This schedule illustrates the order in which major tests are performed and their dependency to each other. For example some tests like unit testing have to be performed before system testing which means system testing depends on unit testing. The diagram below portrays the dependency relationship of the various tests.

Since **unit testing was done** through the development/ coding phase we are left with system and function testing, user acceptance testing, audit testing and system performance testing and which are scheduled as follows.

Subsystem	Type of Testing	Duration (days)	Start Date	End Date
Add Guest Booking	Unit	2	07/10/2012	07/10/2012
	Function		07/10/2012	07/10/2012
	Audit		08/10/2012	08/10/2012
	Performance		08/10/2012	08/10/2012
Cancel Guest Booking	Unit	2	09/10/2012	09/10/2012
	Function		09/10/2012	09/10/2012
	Audit		10/10/2012	10/10/2012
	Performance		10/10/2012	10/10/2012
Change Guest Booking	Unit	2	11/10/2012	11/10/2012
	Function		11/10/2012	11/10/2012
	Audit		12/10/202	12/10/2012

	Performance		12/10/2012	12/10/2012
Guest Booking Enquiry	Unit	2	13/10/2012	13/10/2012
	Function		13/10/2012	13/10/2012
	Audit		14/10/2012	14/10/2012
	Performance		14/10/2012	14/10/2012
Report Generation	Unit	2	15/10/2012	15/10/2012
	Function		15/10/2012	15/10/2012
	Audit		16/10/2012	16/10/2012
	Performance		16/10/2012	16/10/2012

We are now ready to move on to **user acceptance testing** which will conclude the design phase ushering in the implementation phase discussed in the implementation plan outlined earlier.