ICT Engineering

SEP I1
Process Report

Submitted to:

Alexis Claire Wolhovd

Allan Henriksen

Prepared by:

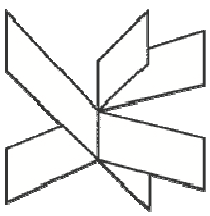
Filip Zelenika 239987

Dean Bozić 239993

Sujan Varma 240135

Anton Akov 241510

Submitted: 15 December 2015



VIA University College

Contents

Group Policy 1

The following contract is for our group to gain a common understanding of our group's goals, ground rules and activities. It is important that each group member abides by the contract. All the points outlined are agreed upon by majority vote**.**

- The primary method of communication will be email and the Facebook Group SEP1 Project, the secondary will be telephone. Telephone calls are to be later communicated via mail/ Facebook group to ensure there is no minor misunderstanding with any member and so that it may be communicated to other members if the need arises.
- Weekly schedules will be drawn up outlining the location and times to meet. The schedule will commence from a Monday but will be decided upon the previous Friday.
- We will have two regular meetings on Tuesday and Friday mornings.
- We are to be punctual and prepared to work as outlined in the schedule.
- All meetings are to be entered into with a prior outlined agenda and goals.
- All meetings are to be documented.
- If a member cannot attend a meeting he is to inform all the other members of the team and will be responsible for following up on the meeting notes.
- We are to treat each group member with respect and consideration during all tasks.
- Our behaviour should be conducive to progressing in our project and should not be disruptive to other group members.
- I will decide the deadlines for task and honour them.
- I will adhere to the penalty decided by my group members if I do not complete my tasks to the required standard or in a timeous fashion.
- We should have completed the project no later than Saturday, 12th December 2015. Allowing us time to debug.
- Roles will be initially assigned to each group member and I agree to rotate them later.
- I will not hesitate to ask for assistance from my team members.
- I will do my best to make my team members understand the work I carry out.
- If I am ill, I am to notify all my group members and am responsible for catching up with work conducted in my absence.

*N.B. The group policy was signed by all members of the group and submitted in hardcopy on the 9th November 2015.

SWOT Analysis 2

2.1 Group SWOT Analysis

Strengths: Quick thinking, positive working environment, wide variety of skills, ability to innovate.

Weaknesses: Small amount of group work experience, to much discussion without much progress.

Opportunities: Improve of programming skills, get used to group work and having different team roles and improve our English

Threats: Lack of motivation to complete the task, to many extra-activities outside the school.

2.2 Individual SWOT Analysis

Dean Bozic:

Strengths: open-minded, quick-thinker and decent programming knowledge

Weaknesses: Too paranoid, little experience with group work

Opportunities: Learn more about the group work, programming and other cultures

Threats: No any specific threats

Filip Zelenika:

Strengths: encouraging, positive and decent programming knowledge

Weaknesses: lack of leadership, hard to motivate

Opportunities: Learn more about programming, increase motivation for further studies.

Threats: No any specific threats

Anton Akov:

Strengths: excellent programming knowledge and very good at completing tasks

Weaknesses: can be stubborn at times and sometimes lack of communication

Opportunities: Learn more about his other skills and group work

Threats: No any specific threats

Sujan Varma:

Strengths: excellent leader, very broad skill set and lots group work experience

Weaknesses: lack of focus and sometimes late for meetings

Opportunities: Learn more about programming and improve his focus skills

Threats: No any specific threats

Considerations 3

Dean Bozić

As the project period was getting closer, I was a bit excited and pessimistic at the same time, since I've never had a real group work experience. I wasn't sure how should we divide our roles in a fair way or what will be my best approach, since not only I have to do a real group work now, but also work with people from other cultures. Nevertheless, after almost three weeks participating and contributing with 3 other students, I believe I gathered enough experience to create my own opinion about group work, especially with people from other countries. Going back to first week, I believe we started on the right food. We all showed our ambition and strengthens that will help us get through this project as smooth as possible. However, at some point of time, we all started showing our weaknesses at which, I believe, we didn't react as fast or productive as we should have. But, as project period was decreasing, we started working as a real job group and started dividing our roles as we at that point believed was the best, but at the same time tried to keep everyone enrolled in everything, so we all know where and how we stand. Second part of the second week and last week embraced a lot of work, but I personally didn't find it very difficult, since we had a great atmosphere and we were helping where we thought our skills could help. All 4 of participated in every discussion regarding project and we always strived to took the best suggestions. All in all, I would say that this is a great experience for me, since like it or not, I'll have a lot of group works and I first have to improve myself wherever possible and then learn how to adapt to any group work situation. I was mostly satisfied with my group and I hope they think the same for me.

Filip Zelenika

Coming into the project period I was thrilled to embark on this new experience. I was excited to start working with the people in my group, we were all on the same page and we got along in class. The first few days of the project period were focused on setting the layout of the project and determining the requirements. The atmosphere was generally positive, we had a fair share of arguments,but in the end we were all willing to get by and remain focused on our objectives. We all quickly found our rythm and began working on contributing as much as possible with our roles. My role was the checker so I was fairly busy going over everything that we did and making sure it all tied together nicely. We regularly checked how we stood with the overall progress and we made sure that we were on schedule. That sometimes implied that we spend a lot of time working at the University or some other place, but we all agreed it was necessary. As the project period was coming to a close we were all rushing to meet the deadline, but the overall atmosphere was excellent, we agreed quickly on every part of the project that was left due. In the conclusion, I'd like to say how this experience broadened my horizons and made me think outside the box. I didn't have much

group work experience before coming to Denmark, at least not of this type, but I think my group and I really pulled our weight and delivered in the end.

Anton Akov

I was excited about taking on the SEP and I was determined to make a pretty good one. I ended up in a group in which everyone shared that goal. The project proved to be a bigger challenge than I expected because at first I didn't have a good understanding of how the group work would go. Personally time was the biggest constraint.

Our group was well rounded. Each member had his own unique skills and was eager to put them to use. We started by planning out the project and had some minor arguments about details certain details but we were always quick to reach a consensus. We quickly got into action started working on the project part by part. We were efficient both working together and separately. Whenever we had questions or needed help by our supervisors or call was answered. My role was mainly turning the ideas we had about the program into a working reality, and I think it turned out rather nicely. As we neared the deadline we did have to rush a couple of things but we stayed focused and on point the entire time.

This project was my first group assignment of this kind that I've participated in. It brought in a good deal of challenges and I ended up learning a lot of valuable things related to both group work and what I did specifically along the way. I'm happy with the way me and my group handled the project and I think it turned out well.

Sujan Varma

I very much looked forward to tackling our semester project as it gave us an opportunity to apply our newly learned skills to a real world environment. When learning programming we learn to program using tasks that are isolated to certain situations whereas the project is very similar but on a much larger scale giving us a global perspective.

I was happy to find a productive and disciplined team. We were all very enthusiastic to prove our talents in programming. I took leadership of the group as I have spent many years studying and doing group work in which much of my time was wasted upon arguing and making decisions as opposed to actually being productive. I led discussions so that they were very in depth and detailed so that every group member was on the same page. The demographic of our group was great to work with. I have studied intensely previously and have work in a corporate environment. Anton had already studied program and Dean and Filip were new a tertiary education environment. The dynamics of our group left me with leadership and delegation of tasks and Anton, being our strongest programmer took lead in programming most of the system. We made the decision to let Anton do most of the code as this was an exercise in a commercial task and putting your best foot forward was key to having a superb system.

In hindsight I realised that leaving most of the programming to one person created other problems regarding the other tasks to be carried out, for example the sequence diagrams and JavaDoc could only be tackled by Anton due to his in-depth knowledge of the code. We soon came to realise that the paperwork would be help back by Anton due to time constraints. We also made the mistake of thinking that it would be fine to work independently at home which proved to be troublesome in terms of communication and unproductive in comparison to working at the university. I have learned to manage my time better and that planning is key to a good system, foresight would be a great quality that was developed in this exercise as well.

Logbook 4

Daily Project Log	
Project Name	SEP1 A15 - The Deer Alley Hotel
Group members	Anton Akov, Sujan Varma, Filip Zelenika and Dean Bozić
Date	Task
6 November 2015	We met as a group for the first time and made the group contract, we also discussed group roles and personal responsibilities.
9 November 2015	Main task for this meeting was determining the deadlines for various group tasks and general discussion about the project.
10 November 2015	Further discussion of the project, focus on the user case. Scheduling the next meeting.
11 November 2015	General discussion of the project.
25 November 2015	The focus of the meeting were the requirements and distributing the workload to each member.
26 November 2015	Determining the priority requirements, starting the programming part of the project.
27 November 2015	Further discussion of the requirements, agreeing upon the priority ones. Continuous programming of the basic classes.
30 November 2015	Starting work on the Use case diagram for the priority requirements. Programming.
1 December 2015	Working on the Use case descriptions for the finished diagrams. Programming
2 December 2015	Work on the Activity diagrams in Astah.
3 December 2015	Continued work on the Activity diagrams, further programming.
4 December 2015	
7 December 2015	Group discussion about the best GUI design.
8 December 2015	Complex programming including work on the GUI, search functions, tables etc.
9 December 2015	Each group member worked on SWOT analysis for the group and the individual ones. Personal process reports, various documentation work.
10 December 2015	Finishing up the GUI, testing out final elements of the program, primary focus on determining the layout for the documentation.
11 December 2015	Work on the Class diagrams, Sequence diagram, process report, each member submits their old and new Bloom's forms.
14 December 2015	Finishing up the final things for the project, grouping it together, checking the spelling and grammar. Program is finished.

Group Task Log 5

Group Task Log			
Project Name	SEP1 A15 - The Deer Alley Hotel		
Group members	Anton Akov, Sujan Varma, Filip Zelenika and Dean Bozić		
Task	Assigned To	Status	Completed
Booking Class	Anton Akov	Tested and Working	Yes
Booking Date Class	Anton Akov	Not Completely Working	No
BookingTab Class	Dean Bozic	Tested and Working	Yes
Bookings Class	Anton Akov	Tested and Working	Yes
BookingTablePanel	Anton Akov	Tested and Working	Yes
FileIO Class	Sujan Varma	Tested and Working	Yes
Guest	Filip Zelenika	Tested and Working	Yes
GuestCheckInTable	Anton Akov	Tested and Working	Yes
GuestTab	Dean Bozic	Tested and Working	Yes
GuestTablePanel	Antov Akov	Tested and Working	Yes
HomeTablePanel	Anton Akov	Tested and Working	Yes
MakeBooking	Filip Zelenika	Tested and Working	Yes
HomeTab	Dean Bozic	Tested and Working	Yes
MAIN	Antov Akov	Tested and Working	Yes
MultiLineHeaderRenderer	Anton Akov	Tested and Working	Yes
testmain	Sujan Varma	Tested and Working	Yes
OnExit	Filip Zelenika	Tested and Working	Yes
PopUp	Dean Bozic	Tested and Working	Yes
CheckOut	Anton Akov	Tested and Working	Yes
Room	Anton Akov	Tested and Working	Yes
RoomTab	Dean Bozic	Tested and Working	Yes
RoomTablePanel	Anton Akov	Tested and Working	Yes
CheckIn	Filip Zelenika	Tested and Working	Yes
NoteEdit	Filip Zelenika	Tested and Working	Yes
Cover Page	Sujan Varma	Approved by Group	Yes
Requirements	Whole Group	Approved by Group	Yes
Use Case	Whole Group	Approved by Group	Yes
Use Case Activity Diagrams	Dean and Filip	Approved by Group	Yes
Contents	Sujan Varma	Approved by Group	Yes
Daily Log	Filip Zelenika	Approved by Group	Yes
SWOT	Whole Group	Approved by Group	Yes
Use Case Description	Sujan and Dean	Approved by Group	Yes
Case Diagrams	Filip and Anton	Approved by Group	Yes

Group Roles Description	Filip Zelenika	Approved by Group	Yes
Task Log	Dean Bozic	Approved by Group	Yes
Reflections on Outcome	Whole Group	Approved by Group	Yes
Meetings Report	Sujan Varma	Approved by Group	Yes
Blooms Profile	Whole Group	Approved by Group	Yes
Introduction	Sujan Varma	Approved by Group	Yes
Sequence Diagrams	Sujan and Anton	Approved by Group	Yes
GUI Explanation	Dean and Filip	Approved by Group	Yes
MakeBooking Class Explanation	Anton Akov	Approved by Group	Yes
Activtiy Diagrams Description	Whole Group	Approved by Group	Yes

Bloom's Forms 7

Dean Bozić

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom' s level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6															
	5															
Good	4										▮	▮	▮	▮		
	3														▮	
	2	▮														▮
Basic	1		▮	▮		▮		▮	▮							
No knowledge	0				x		x			x						

Table 1

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom' s level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6															
	5										▮		▮	▮		
Good	4	▮				▮	▮					▮			▮	▮
	3		▮					▮	▮							
	2			▮												
Basic	1									▮						
No knowledge	0															

Table 2

Filip Zelenika

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....14.9.2015.....	Bloom' s level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6											X				
	5															
Good	4										▮		▮	▮		
	3															
	2					▮	▮	▮							▮	▮
Basic	1	▮		▮					▮							
No knowledge	0		X		X					X						

Table 3

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6											x				
	5							x					x	x	x	
Good	4			x		x	x									x
	3										x					
	2		x						x							
Basic	1	x								x						
No knowledge	0				X											

Table 4

Sujan Varma

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6										X	X		X		
	5			x									x		X	
Good	4	x			x					x						x
	3		x				x	x	x							
	2					x										
Basic	1															
No knowledge	0															

Table 5

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom's level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6										X	X		X		
	5	x		x				x					x		X	X
Good	4		x		x	x	x		x	x						
	3															
	2															
Basic	1															
No knowledge	0															

Table 6

Anton Akov

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom´s level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6			x			x									
	5		x								x	x		x		
Good	4					x		x					x		x	x
	3								x	x						
	2	x														
Basic	1				x											
No knowledge	0															

Fill in this form – include it in your portfolio – discuss it with the rest of the group Date.....	Bloom´s level	Keeping a portfolio	Reflecting on learning	System development	SCRUM	Java Programming	Object-oriented design and programming	UML	Web Programming	Database design	Written English	Spoken English	Team working	Sharing knowledge	Project planning	Presentation / exam skills
Excellent	6			x			x									
	5		x			x					x	x				
Good	4							x	x	x			x	x	x	x
	3															
	2															
Basic	1	x			x											
No knowledge	0															

ICT Engineering

SEP I1 A15
Project Report

Submitted to:

Alexis Claire Wolhovd

Allan Henriksen

Prepared by:

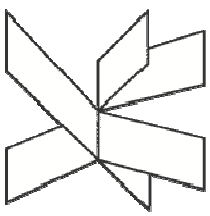
Filip Zelenika 239987

Dean Bozić 239993

Sujan Varma 240135

Anton Akov 241510

Submitted: 15 December 2015



VIA University College

Contents

Introduction 1

The project presented to us was based on a small, 15th century hotel having problems with their manual booking system. Due to its heritage and atmosphere the owner was reluctant to add any modernisation but due to numerous problems, had to. An interview was conducted with the owner of the hotel to establish the problems, priority requirements and additional requirements of the system to be designed. We identified four major problems experienced by the manual booking system i.e. double bookings, not being able to view available rooms, unsaved bookings and viewing daily check-ins and check-outs. The problems faced, outlined our challenges when designing the system. The system had to provide a user friendly graphical interface, minimising human error while facilitating all the requirements of the hotel's bookings. The details of the system we developed are stated in the followings pages of this report.

Project Analysis 2

1. The system must be able to store guest name, arrival/departure date and room number for the booking.

2. User must be able to search through all the bookings based on guest name or room number in order to check in a guest if he made a booking.

3. User should be able to register the guest using the guest's name, home address, phone, e-mail, passport number, date of birth and the arrival/departure date and room number at check-in.

4. The system must be able to register a departure date and calculate the price at check out.

5. The system must be able to display the expected arrivals and departures for any given day.

6. The user should be able to search through all the bookings based on arrival/departure date in order to edit/cancel a booking.

7. The system must be able to display all the available rooms based on time period, price range and room type.

8. The system should be able to store the price information for all of the rooms.

9. The system should store the following information about the rooms: type, number, price and the dates it is booked for.

10. The system must be able to add additional services to the final price.

11. The system should display a message when the user increases the price of the rooms.

12. The user should be able to decrease the price of the rooms.

13. The system must have an option to add an extra bed when registering the booking.

14. The system should notify the user if it's after 18.00 and the guest hasn't checked in for a room.

15. The user should be able to edit the regular price of the rooms.

2.3 Priority Use Case Diagram

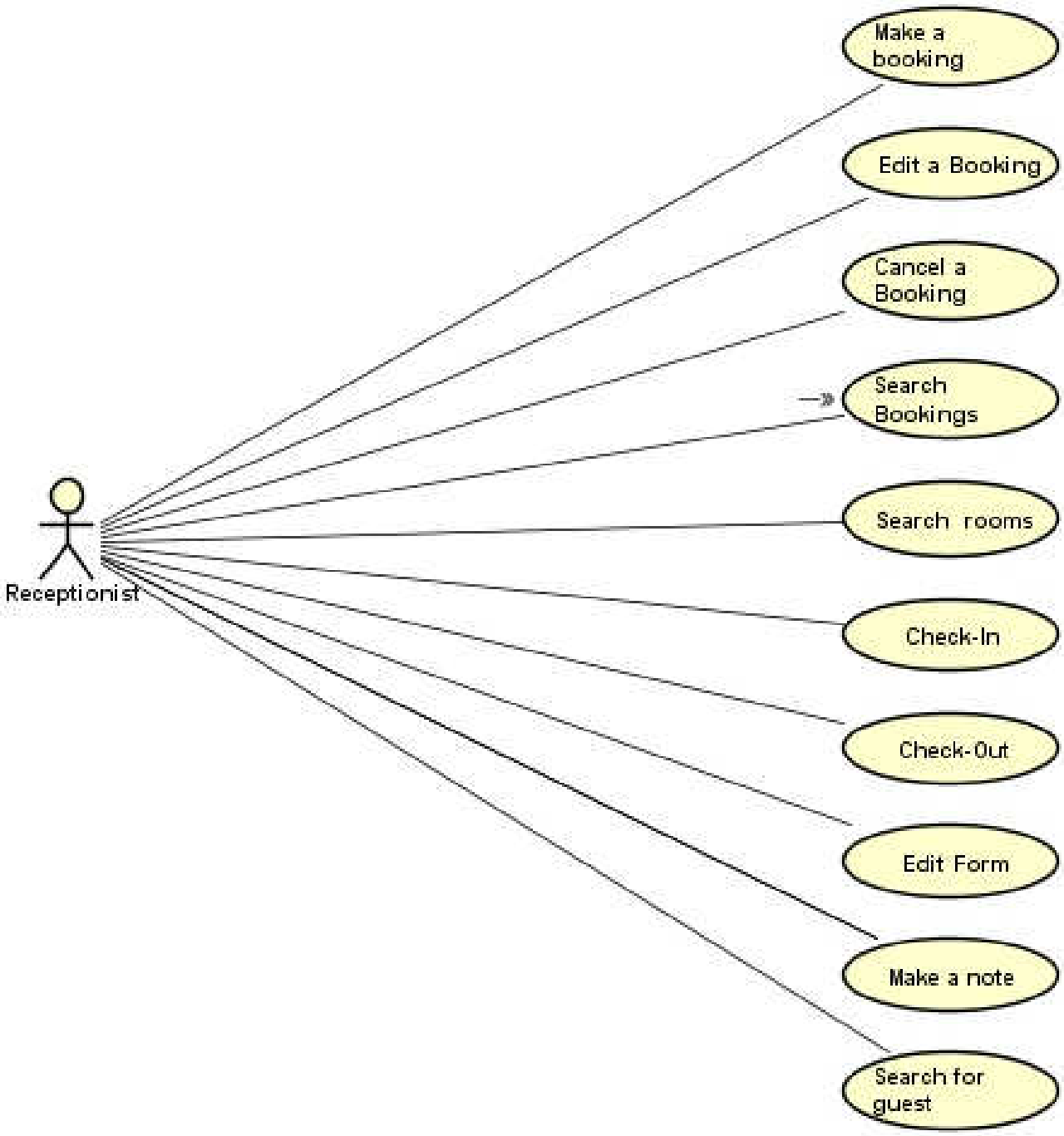


Figure 1: Priority Use Case Diagram

Figure 1 illustrates the priority use cases performed by the receptionist of the hotel. It does not included all the use cases encountered by the receptionist but merely the essential use cases required for the operation of the booking system.

2.4 Use Case Description – Create a booking

Use Case	Create a Booking
Summary	The system must be able to store guest name, arrival/departure date and room number for the booking.
Actor	Receptionist
Precondition	A room must be available.
Post condition	A booking must be created.
Base Sequence	1) Search available rooms. 2) Click 'Make a booking' button. 3) System displays a booking form. 4) Actor populates booking form. 5) Actor clicks on 'Save' button. 6) System prompts user for confirmation. 7) If yes, system creates and saves the booking.
Exception Sequence	User clicks "No": 1-6 as base sequence 7.System redirects user to matched bookings use case end
Note	

Table 1

Table 1 displays the use case description for creating a booking. Creating a booking is the most prolific class in the system and we have chosen this class to demonstrate the methods we applied to complete this project. The documentation for all other classes and use cases can be found in the appendices. The following pages of the documentation will be orientated around the 'Create a booking' use case as well.

2.5 Activity Diagram – Create a booking

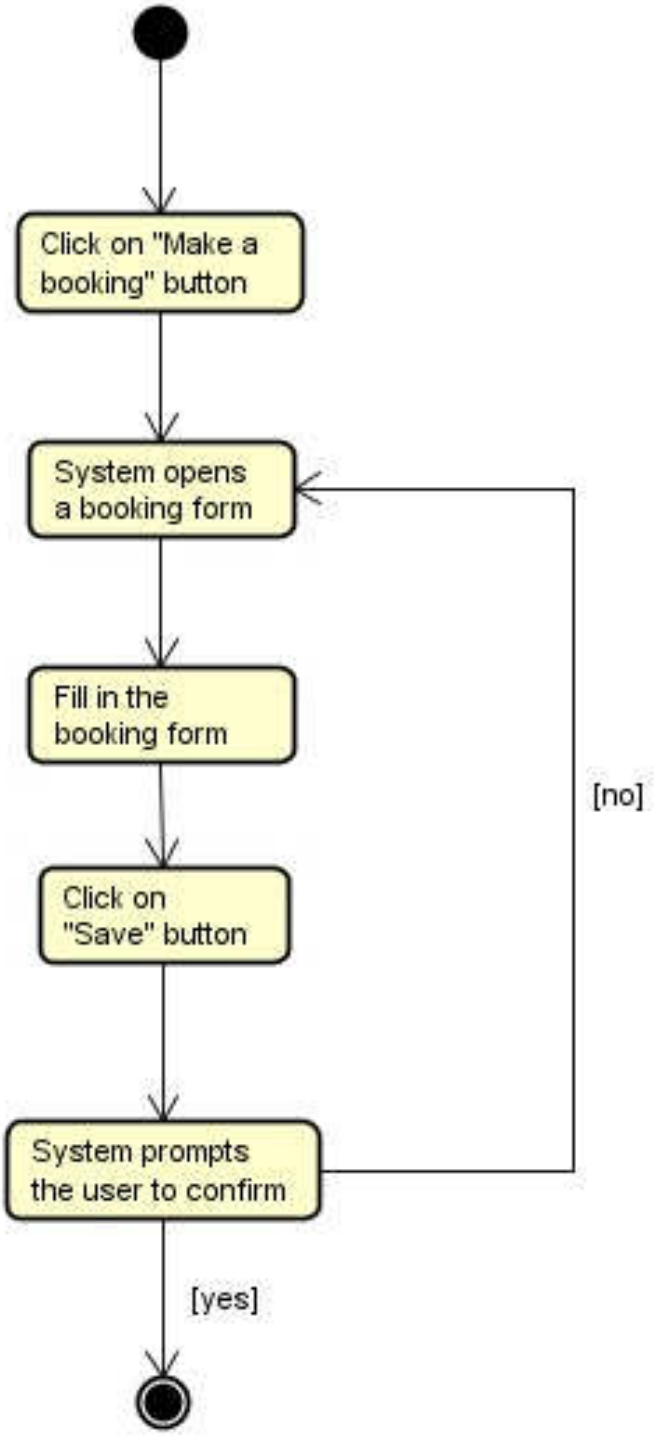


Figure 2

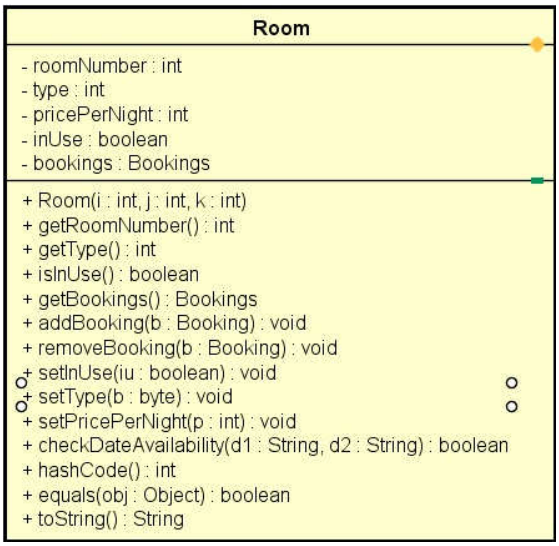
Figure 2 illustrates the activity diagram for the 'Create a booking' use case.

Design 3

3.1 UML Class Diagram (Condensed)

3.2 UML Class Diagram (Expanded)

3.3 UML Class Diagram Breakdown



Room class takes care of all functions that are needed for a searching a room action. Further, it also takes information from a booking class and uses it as a connection between classes. This class stores information about room number, type and price. In addition, it has a method called checkDateAvailability() which has a function to check if wanted room is available at a certain date. Room class is also used by a Rooms class.

Booking class takes information from Room, Guest and BookingDate classes and incorporates it all in one. It stores an array of Room class and uses it for displaying booked rooms by simply using getters and setters. Also, it takes information from BookingDate class and uses it in isBookingToday() method to check if there are any particular ongoing bookings at that day. Finally, with using information from guest class it checks if the guest is already checked in.

3.3 GUI Design Breakdown

The main focus for us when making the GUI was usability. We decided that the program should be simple to use for everyone, regardless of their computer skills. The overall design is very neat and simplistic, there are no buttons or fields that are out of place and we believe it should be very easy to find the exact function that the user needs.

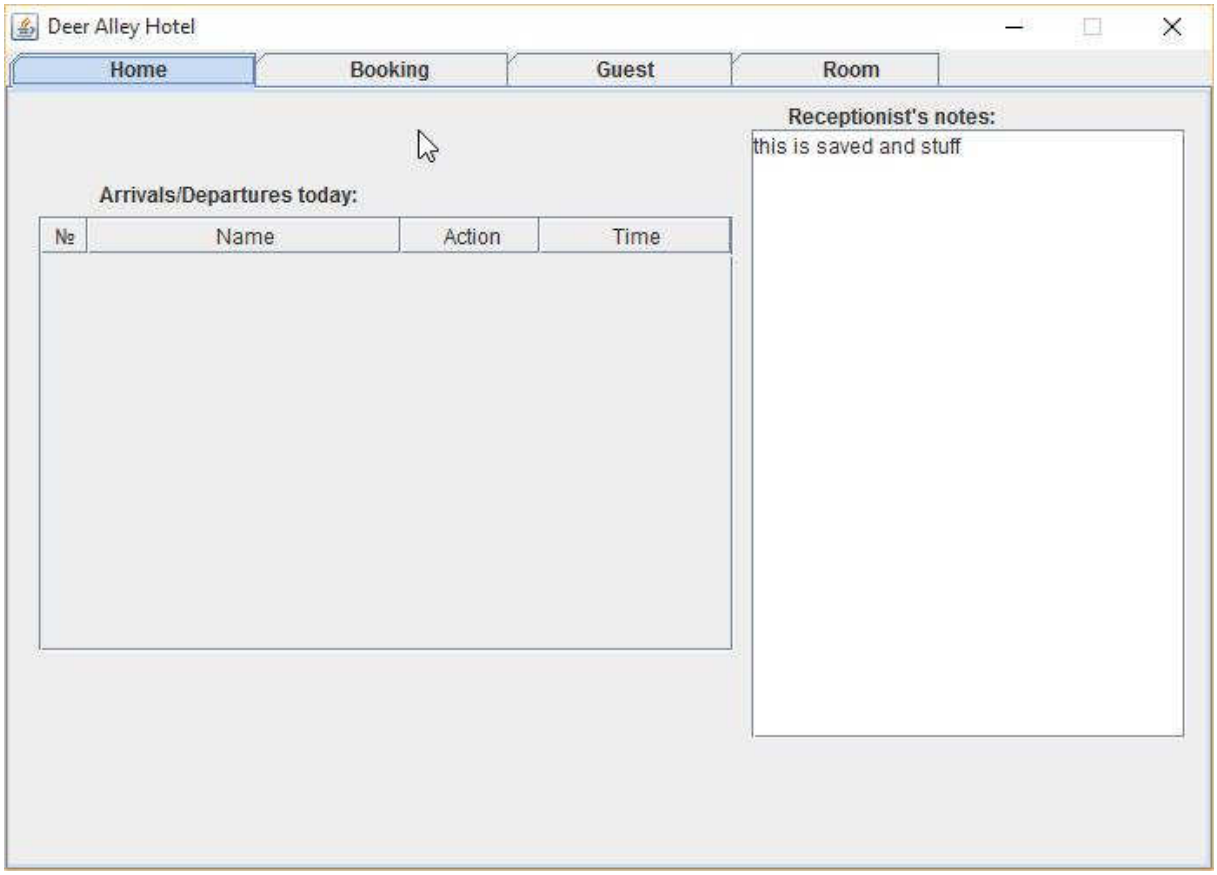


Figure 3 The Home tab

The home tab features two important features:

1. A daily schedule that displays all of the check-in and check-outs for that given day. The schedule features everything that the user needs, it displays the exact time of the check-in/check-out, the name of the person that made the booking and the number of the room in question.

2. A notepad field where the user can type in notes by simply clicking on the notepad field and writing. We implemented an autosave function, so every note is saved on exit. We believe that being able to create and view notes is very important to have while working at the receptionist desk.

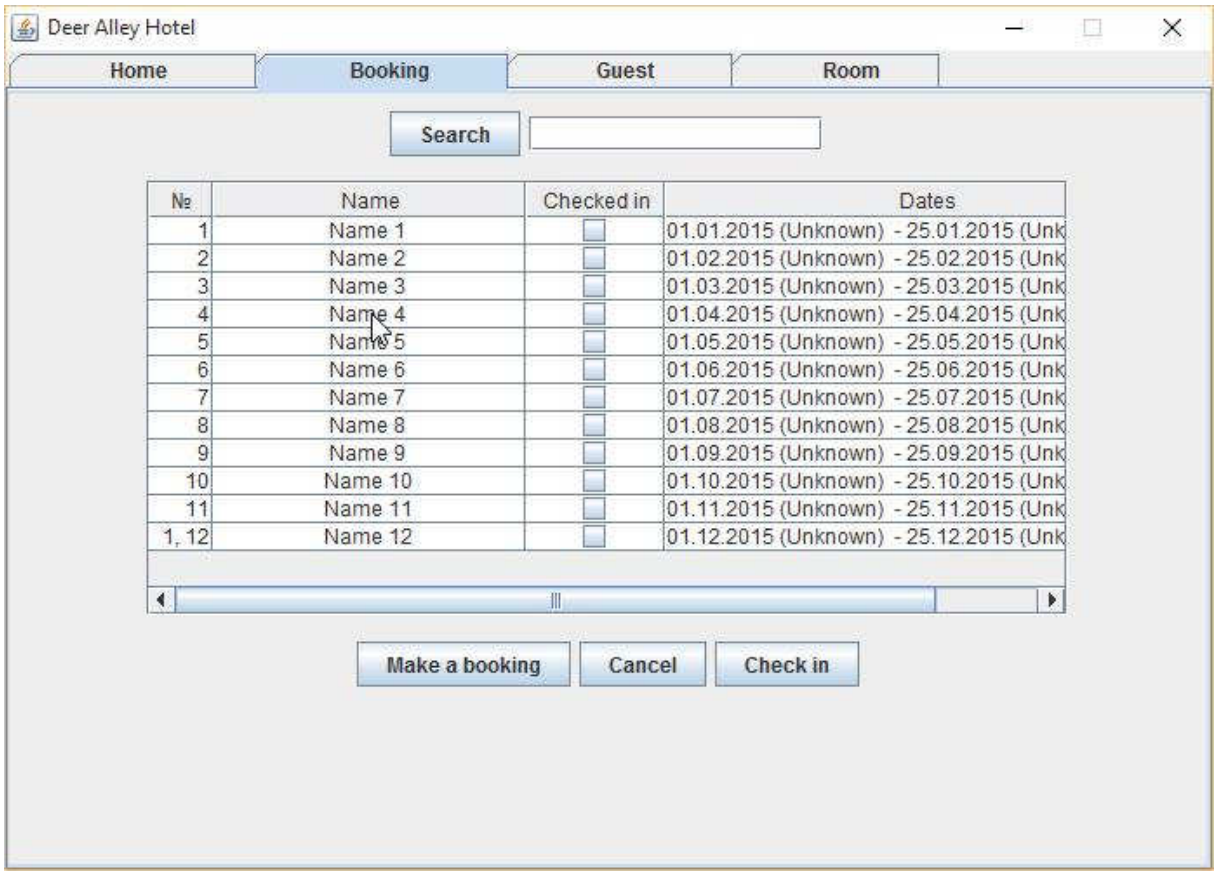


Figure 4 The Booking tab

The layout of the booking tab features a search function where the user can search for a booking by any of the given parameters (name of the person that made the booking number of the room or by date). It also features a table that displays all the bookings and also fills with the appropriate bookings when the user searches. At the bottom there are three buttons that allow the user to make/cancel a booking and also check in the guests for any given booking. Make a booking button opens a new form with all the information for a booking and saves the booking. Cancel a booking simply cancels the selected booking after a prompt.

Check in button opens a new form when clicked. The form features a new table and several options to check in the guests for the wanted booking and also a notepad where the user can make notes about that booking.

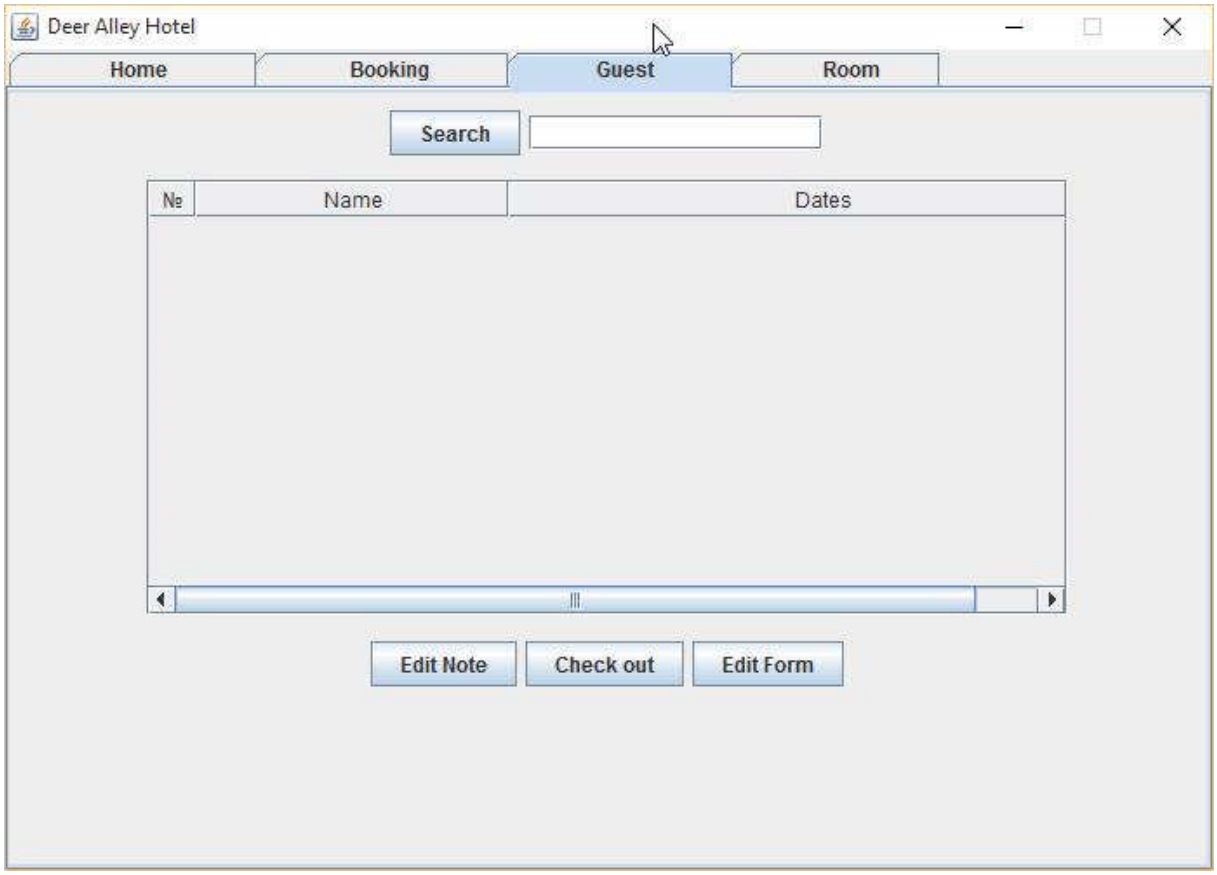


Figure 4 The Guest Tab

The guest tab also has a search function that is able to display all the guests that match the parameters the user searched by (the user is able to search by any of the check in parameters such as name, passport number, phone number etc.). It also features another table with the guest details. The user is also able to make notes about the guests, edit the check in forms and also check out the guest. When the check-out button is clicked, it opens a new window with

the guest notes and our system calculates the price of the stay for that guest. We made the check-out process as simple as possible.

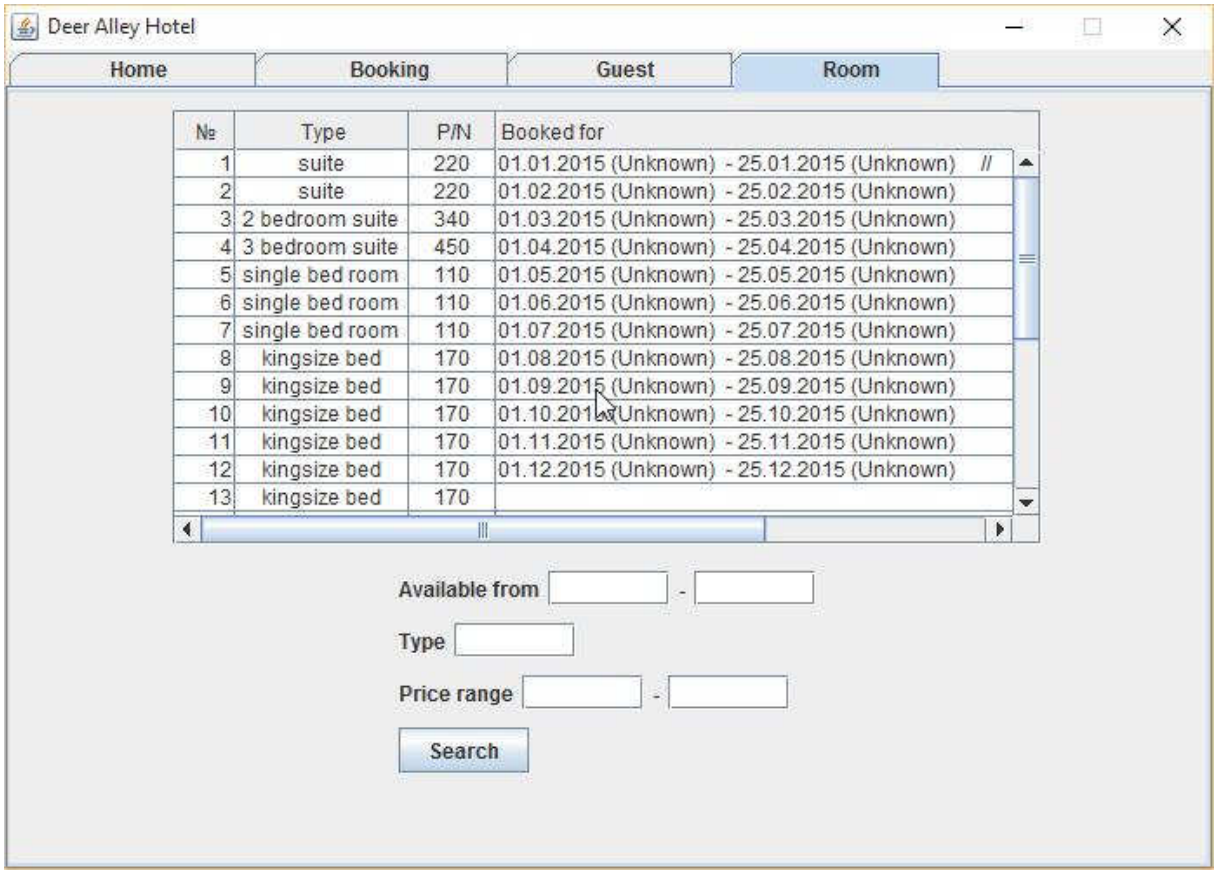


Figure 5 The Room tab

Our program also features the room tab that displays all the rooms and specifies the price per night, the type of the room and also whether it is booked and if so for how long. The user is also able to search for rooms based on availability on any given date, type of the room and price range. The table is automatically populated with the matching rooms for those parameters giving the user easy access to important information.

3.4 Sequence Diagram

Implementation 4

We would like to highlight the way we selected to display the information in the system we developed for this project. We chose to use tables because they provided us with a large array of functionality (information can be selected, filtered, sorted easily) and is easy to use by the user to a large extent. Tables are included in the better part of the windows in the program, to give an example of how we realized that we'll show the source code behind the one in the booking tab.

Code Snippet 1:

```
public class BookingTablePanel extends JPanel
{
```

Each table's main container is a JPanel. The reason for that is that panels are easy to integrate into the rest of the GUI.

Code Snippet 2:

```
public BookingTablePanel(Bookings b, JButton searchB, JTextField searchT, JButton mB
{
    text = searchT;
    butt = searchB;

    newB = mBooking;
    cancelB = cancel;
    checkinB = checkin;
    editB = edit;

    bookings = b;
    dates = 0;
    rooms = 0;

    listener = new ButtonListener();
    butt.addActionListener(listener);
    newB.addActionListener(listener);
    cancelB.addActionListener(listener);
    checkinB.addActionListener(listener);
    editB.addActionListener(listener);

    thispanel=this;

    construct();
}
```

The action listener for the buttons is coded inside of the panels holding the tables, thus the constructors take the information the table holds as arguments plus the buttons that command the actions. The constructors themselves just assign the arguments to variables in the object and do actions that wouldn't need to be repeated later on, i.e. assigning the button listener to the button and this to a variable so it can be used in the action listener to access the object it's in. The construct method is called to create the table. It's not a part of the constructor because on occasion it has to be called more than 1 time.

Code Snippet 3:

```
private void construct()
{
   int i = 0;
   int j = 0;
   String tempRoomString = "";
   String tempDateString = "";
   data = new Object[bookings.getList().size()][5];
   while(i<bookings.getList().size())
   {
      while(j<bookings.get(i).getBookedRooms().length)
      {
         tempRoomString = tempRoomString + bookings.get(i).getBookedRooms()[j].getRoomNumber()+", ";
         j++;
      }
      data[i][0] = tempRoomString.substring(0, tempRoomString.length()-2);
      if(rooms<j)rooms = j;
      j=0;
      tempRoomString = "";


      data[i][1] = bookings.get(i).getName();
      data[i][2] = (boolean)bookings.get(i).isCheckedIn();

      data[i][3] = bookings.get(i).getDates();

      data[i][4] = bookings.get(i);


      i++;
   }
```

The first thing the method does is initialise the 2-dimensional Object array which holds the information for the table. It is sized accordingly to the number of elements it is expected to hold. The first while loop goes through each element and populates the columns in each row with information from it. The nested while loop creates a string containing the room numbers of the rooms that particular booking is reserved for. The 3rd and 5th column are interesting in the sense of the information they hold. The former one contains a Boolean object, which is rendered as a checkbox on the screen and the latter contains the object that the information for the row was acquired from itself. The reason for this is to be able to access the object when a selection is made without having the need to go through to complicated comparisons in order to get to it, which would waste both programming and system time.

Code Snippet 4:

```
model = new DefaultTableModel(data, columns) {
   public Class getColumnClass(int column) {
     Class returnValue;
     if ((column >= 0) && (column < getColumnCount())) {
       returnValue = getValueAt(0, column).getClass();
     } else {
       returnValue = Object.class;
     }
     return returnValue;
   }
   @Override
   public boolean isCellEditable(int row, int column) {
       //Only the third column
       return false;
   }
 };

table = new JTable(model);
```

The table is created by using a default table model instead of having the information directly fed to it as this method enables us to format the columns appearance and give it the desired look.

Code Snippet 5:

```
DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
centerRenderer.setHorizontalAlignment( JLabel.CENTER );

DefaultTableCellRenderer rightRenderer = new DefaultTableCellRenderer();
rightRenderer.setHorizontalAlignment( JLabel.RIGHT );
```

Simple cell renderers that format the alignment of the information in the cells they're assigned to.

Code Snippet 6:

```
TableColumn column;
table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
column = null;
for(int ii = 0;ii<5;ii++)
{
   column = table.getColumnModel().getColumn(ii);
   switch(ii)
   {
      case 0 : column.setPreferredWidth(32+(rooms-1)*5);column.setCellRenderer(rightRenderer);break;
      case 1 : column.setPreferredWidth(180);column.setCellRenderer(centerRenderer);break;
      case 2 : column.setPreferredWidth(80);break;
      case 3 : column.setMinWidth(300);break;
      case 4 : table.getColumnModel().removeColumn(column);
   }
}
```

This particular part of the code is used to adjust the width and format the alignment of the visible columns. The last column containing the object is removed from the table model and

isn't present in the visible part but the access to the information is preserved and still accessible trough the getValueAt(row, column) method.

Code Snippet 7:

```
sorter = new TableRowSorter<TableModel>(model);
table.setRowSorter(sorter);
```

Code creating a table row sorter specific for the current table model and assigning it to the table. Doing this allows the table to be sorted by clicking on the column headers and be easily filtered later on.

Code Snippet 8:

```
pane = new JScrollPane(table,JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED, JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
add(pane, BorderLayout.CENTER);

pane.setPreferredSize(new Dimension(530,250));
```

New JScrollPane created to hoist the table and allow for it to be scrollable and user friendly. It is then added to the JPanel holding the whole thing together and has its preferred size set. We are setting the size on the pane and not on the panel instead because if done the other way around it doesn't work, as the pane viewport size is calculated wrong.

Code Snippet 9:

```
public void remake()
{
   remove(pane);
   construct();
   validate();
}
public void updateMain()
{
   MAIN mainW = (MAIN) SwingUtilities.getWindowAncestor(this);
   mainW.remakeAllTables();
}
```

Two other methods vital to the smooth functionality of the table. The updateMain methods gets a reference to the main window which is holding all the tabs using the getWindowAncestor method in the SwingUtilities package and then calls the remakeAllTables method which triggers a chain of methods which goes into each tab and triggers the remake method for each table. The updateMain method is called whenever a change into the table information is made and the tables need to be updated accordingly. The remake method removes the pane and calls the construct method for each table. The pane

needs to be removed and replaced because if a different number of rows is present after the change the pane will expect a bigger table to display and result in a unstructured looking bug occuring when an attempt is made to scroll to the expected place of the bottom rows. The validate method is called to trigger a refresh on the current screen and display the new table.

Getting to the action listeners, each of them is separated in ‘if’ block statements serving each different button.

Code Snippet 10:

```
private class ButtonListener implements ActionListener
{
   public void actionPerformed(ActionEvent e)
   {
      if(e.getSource() == butt)
      {
         String txt = text.getText();
         if (txt.length() == 0) {
            sorter.setRowFilter(null);
         } else {
            sorter.setRowFilter(RowFilter.regexFilter(txt));
         }
      }
      if(e.getSource() == newB)
      {
         new MakeBooking(bookings, (BookingTablePanel)thispanel);
      }
```

The first ‘if ‘block is for the Search button. It gets the text from the text field above the table and manipulates the table row filter created earlier, according to it. The second block is for Make booking button. The first argument is the bookings list, the created booking is added to and the second one is the variable which allows the MakeBooking to access the current JPanel holding the table and call updateMain after the booking is created to update the tables in the program.

Code Snippet 11:

```
if(e.getSource() == cancelB)
{
   int tablerow = table.getSelectedRow();
   tablerow = table.convertRowIndexToModel(tablerow);
   Booking selected = (Booking) table.getModel().getValueAt(tablerow, 4);

   int choise = JOptionPane.showConfirmDialog(null, "Are you sure you want to dele
   if(choise == JOptionPane.YES_OPTION)
   {
      bookings.delete(selected);
      updateMain();

      //getParent().getParent().
   }
}
```

The first 3 lines in the block for the Cancel button are used to access the Booking object stored in the 5th, invisible column of the table. ConverRowIndexToModel is used because the getSelectedRow returns the index of the visible row selected. Using just the latter method is insufficient because if the table's visible part has been sorted/filtered the visible rows' order wouldn't match the order in the table model which holds the information. After the index is converted to point to the correct row in the model the object for the selected row is obtained. A Yes/No choice is given to the receptionist to confirm the deletion of the booking. All tables are updated after deletion.

Code Snippet 12:

```
if(e.getSource() == checkinB)
{
   int tablerow = table.getSelectedRow();
   tablerow = table.convertRowIndexToModel(tablerow);
   Booking selected = (Booking) table.getModel().getValueAt(tablerow, 4);

   if(!selected.isCheckedIn())new CheckIn(selected, bookings, (BookingTablePanel) thispanel);
}
if(e.getSource() == editB)
{
   int tablerow = table.getSelectedRow();
   tablerow = table.convertRowIndexToModel(tablerow);
   Booking selected = (Booking) table.getModel().getValueAt(tablerow, 4);

   if(!selected.isCheckedIn())new MakeBooking(bookings, (BookingTablePanel)thispanel, selected);


}
```

The last two ‘if’ blocks are for the Check in booking and Edit booking buttons respectively. The same piece of code is used to access the selected object. If it’s not been checked in already, the corresponding windows are created. The constructor for the MakeBooking which takes a booking as an extra argument populates it’s fields with information from it, to make editing easier.

System Test 5

We rigorously beta tested our system in accordance with our priority use cases to ensure that all the functionality worked as intended. The following tables list the functionality tested and the results of the testing, under the heading of the use case that applies to.

1. Cancel a Booking

1) Actor enters booking information.	YES
2) System uses the information to search against all bookings.	YES
3) System displays all bookings matching the reference information.	YES
4) The actor selects the correct booking and the booking form is displayed by the system.	YES
5) The actor can then cancel the booking.	YES
6) System prompts the user for confirmation.	YES
7) User confirms.	YES

2. Check-In

1) If guest has a booking, search bookings using booking info.	YES
2) Actor selects the correct booking.	YES
3) Actor clicks on 'Check-in' Button.	YES
4) System displays check in form.	YES
5) Actor populates check-in form and clicks the save button.	YES
6) System prompts actor for confirmation or cancellation.	YES
7) If cancelled system opens a new check-in form and if saved system returns to home screen.	YES
8) If guest does not have a booking, follow the steps to create a booking and continue through steps 1 -8 mentioned above.	YES
9) System prompts the user	YES
10) User confirms	YES

3. Check-Out

1) If guest has a booking, search bookings using booking info.	YES
2) Actor selects the correct booking.	YES
3) Actor clicks on 'Check-out' Button.	YES
4) Actor clicks on "Check-out" Button	YES
5) System prompts actor for confirmation or cancellation.	YES
6) Actor confirms check-out and the system deletes the booking.	YES

4. Edit a Booking

1) Actor enters booking information.	YES
2) System uses the information to search against all bookings.	YES
3) System displays all bookings matching the reference information.	YES
4) The actor selects the correct booking and the booking form is displayed by the system.	YES
5) The actor can then edit the booking form.	YES
6) The actor clicks 'Save'.	YES
7) The system prompts the user for confirmation or cancellation.	YES
8) The actor clicks 'OK' and the system saves the new information to the booking form.	YES

5. Edit Form

1) Actor enters guest information.	YES
2) System uses the information to search against all guests.	YES
3) System displays all guest matching the reference information.	YES
4) The actor selects the guest and the check in form is displayed by the system.	YES
5) The actor can then edit the guest information.	YES
6) System prompts the user	YES
7) User confirms	YES

6. Make a Booking

1) Search available rooms.	YES
2) Click 'Make a booking' button.	YES
3) System displays a booking form.	YES
4) Actor populates booking form.	YES
5) Actor clicks on 'Save' button.	YES
6) System prompts user for confirmation.	YES
7) If yes, system creates and saves the booking.	YES

7. Edit a Note

1) Actor enters guest information.	YES
2) System uses the information to search against all guests.	YES
3) System displays all guests matching the reference information.	YES
4) The actor selects the guests and the clicks the "Make a note" button.	YES
5) The actor can add the note	YES

8. Search a Room

(1) Actor populates the search fields.	YES
(2)Actor clicks on search button.	YES
(3)System displays all available rooms matching search requirements.	YES

9. Search the Guest

1) Actor enters booking information.	YES
2) System uses the information to search against all bookings.	YES
3) System displays all bookings matching the reference information.	YES

10. Search a Booking

1) Actor enters guest information.	YES
2) System uses the information to search against all guests.	YES
3) System displays all guests matching the reference information.	YES

Results 6

1. The system must be able to store guest name, arrival/departure date and room number for the booking.	Working
2. User must be able to search through all the bookings based on guest name or room number in order to check in a guest if he made a booking.	Working
3. User should be able to register the guest using the guest's name, home address, phone, e-mail, passport number, date of birth and the arrival/departure date and room number at check-in.	Working
4. The system must be able to register a departure date and calculate the price at check out	Working
5. The system must be able to display the expected arrivals and departures for any given day	Working
6. The user should be able to search through all the bookings based on arrival/departure date in order to edit/cancel a booking	Working
7. The system must be able to display all the available rooms based on time period, price range and room type.	Working
8. The system should be able to store the price information for all of the rooms.	Working
9. The system should store the following information about the rooms: type, number, price and the dates it is booked for.	Working
10. The system must be able to add additional services to the final price	Working
11. The system should display a message when the user increases the price of the rooms.	Not implemented
12. The user should be able to decrease the price of the rooms.	Not implemented
13. The system must have an option to add an extra bed when registering the booking.	Not implemented
14. The system should notify the user if it's after 18.00 and the guest hasn't checked in for a room.	Not implemented
15. The user should be able to edit the regular price of the rooms.	Not implemented

Discussion 7

The greatest hurdle to designing our system was not knowing our limitations in regards to coding, opposed to how we would plan the GUI to be as user friendly as possible. The core of the system that we developed is based on the user requirements, generated from the interview with the owner. Considering that the hotel is a 15th century, unmodern establishment, the likelihood of staff not being computer savvy is higher. We have designed our GUI with this in mind and have made it very user friendly. This would also make it less of a task to train new staff on the system's operation. We have minimised the number of confirmations to perform actions to only the most important functions to reduce the annoyance of repetitive, mundane actions which create a frustrating user experience. We have incorporated every priority use case into the system developed. We have included some additional user requirements and excluded some due to time constraints and to have a balanced project in regards to coding and documentation.

Conclusion 8

As can be seen from the program, we met our aim, which was successful implementation of all priority requirements. Further, all requirements have been specifically analysed and tested, however, not every has been implemented due to importance order and time limit. Having in mind that program is supposed to be user-friendly, every use case description and activity diagram were particularly checked and compared to a program. This software is undoubtedly comprehensive as well as fast and doesn't require any specific computer knowledge. All in all, it allows receptionist to make a booking, check in/out guest much easier and have much better surveillance of all ongoing events and actions.

DEAN THE CONCLUSION SEEMS A BIT VAGUE AND CAN BE APPLIED TO ANYONES PROJECT REPORT. CAN WE HAVE SOME SPECIFICS AND ELLABORTATE

References 9

Programming Tutorials and Source Code examples,2015

Available from: http://www.java2s.com/Tutorial/Java/CatalogJava.htm [25 February 2015].

Oracle and/or its affiliates,2015

Oracle Java Help Center.

Available from: < https://docs.oracle.com/en/java/> [3 December 2015].

Stack Exchange Inc,2015

Stack Overflow.

Available from:< http://stackoverflow.com/> [10 December 2015].

PLEASE LOOK AT THE HARVARD REFERNCE SYSTEM AGAIN