Hotel System with Java and MySQL

Veikko Pukkila

Bachelor's Thesis
DP in Business Information Technology
2009

Author Veikko Pukkila	**Group** BITE06SX
The title of your thesis Hotel System with Java and MySQL	**Number of pages and appendices** 55 + 1
Supervisors Arvo Lipitsäinen	

This thesis is a system work for a Hotel, which required a reservation system for their customers. The thesis project was split into parts and the scope was determined to specify which services were needed.

The first part is the definition and requirements of the system, which contains information about the services and data of the system. Diagrams are included to make the documentation clearer. Services belong to two main groups, reservation services and administration services.

The second part focuses on the design of the system, mainly user interfaces and the database model. Also there are images of possible solutions and a diagram to specify the database structure. The database is designed for the use of all the defined services.

The third part is about the implementation and testing of the system when it was being developed and finished. There are images to show the solutions that have been used. The implemented services are for creating a reservation and confirming a reservation.

Finally there is a sum-up and a report of the overall process of development and learning. The working hours during the project are reported in a table with the information of what has been done.

Key words
Java, MySQL, Hotel, Reservation

Abbreviations

HTML	Hypertext Markup Language
JSP	Java Server Pages
http	Hypertext Transfer Protocol
MVC	Model, View, Controller
JSTL	JavaServer Pages Standard Tag Library
Ajax	Asynchronous JavaScript and XML
SQL	Structured Query Language

Table of contents

1 Introduction

This thesis is based on the needs of a hotel requiring a reservation system. The sponsor of the thesis was interested in having an on-line system to handle their room reservations. In order to make the work big enough for a thesis, other possible requirements and services were specified into the project scope. Because of this the main part of the thesis is on the definition of services and implementation of the key reservation services. The project steering meetings were held on-line with Skype calls and Google Documents as a tool for communication. The implementation of the system was done on Linux work stations with Eclipse IDE as development tool and MySQL database with phpMyAdmin-administration tool.

2 Definition and Requirements of the System

The purpose of this part is to clarify the over all system definition to the client and the designers of the system. This documentation is based on the needs of the hotel system which in turn are based on interviews with the hotel manager and a Skype conference with the thesis advisor.

This part focuses on managing data in the Hotel System. All services are included.

3 System Overview

This part of the document provides an overall description of the Hotel System, describing the use cases and actors.

3.1 Overall Description

The Hotel System will be a part of the user’s business operations. The system is used for storing and managing hotel’s data regarding different kind of reservations and managing the hotel room situation. The main services of the system are to create new reservations for hotel rooms, the hotel restaurant or tables in the restaurant. The system is planned for two kinds of users, which have different access rights to the services and data of the system. (Interview, Maritta Lehtinen.)

3.2 Use Case Diagram

Hotel System Use Case Diagram with Dependencies

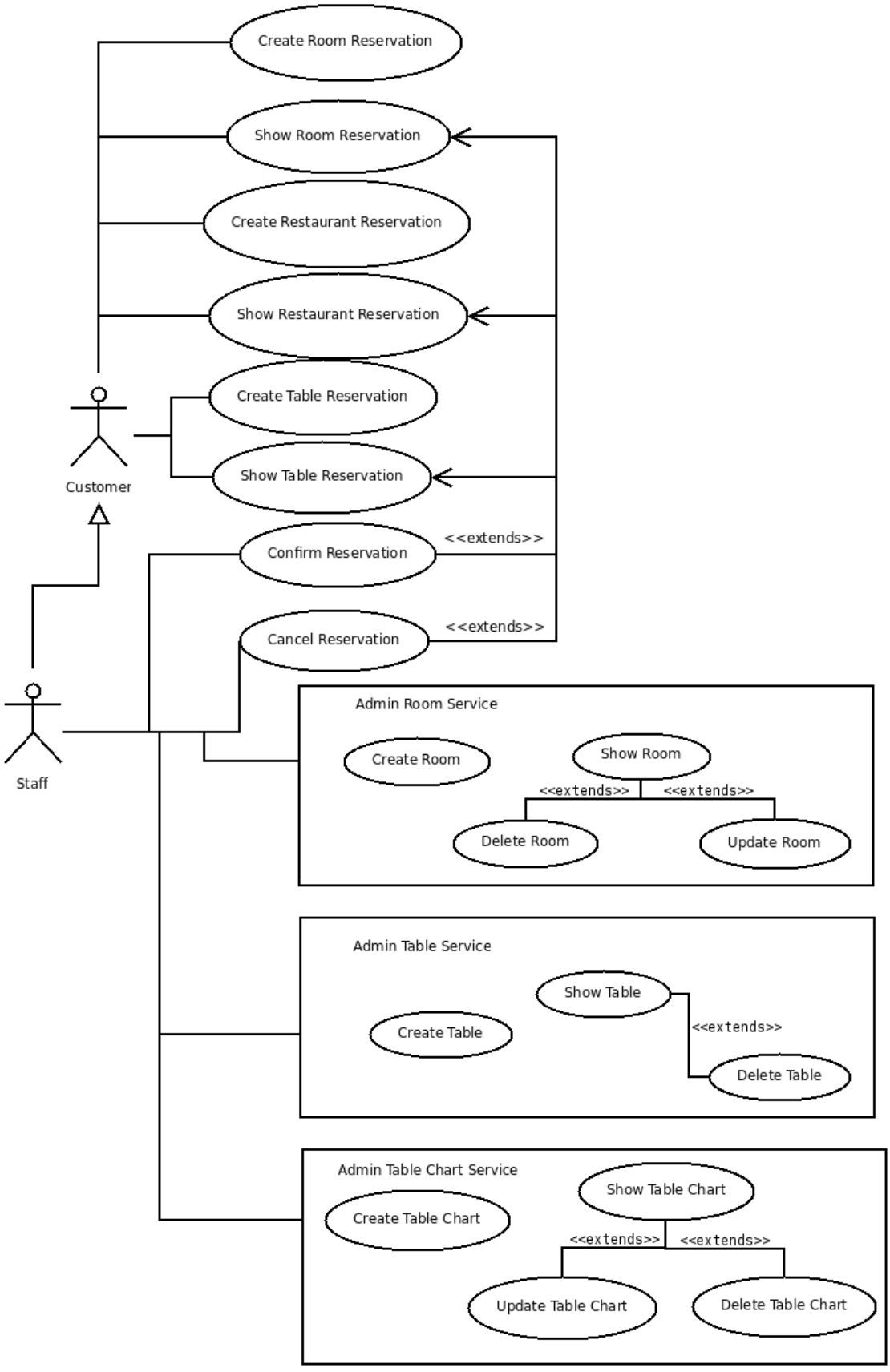

Figure 1. Use Case Diagram for the System

3.3 Services

Here are listed the above mentioned use cases as services with slightly different names uniting the reservation and instance services under one as they serve a similar service. Later in the Use Case descriptions are specified the necessary data for each service variation (use case).

3.3.1 Reservation Services

Create Reservation
This function stores a new reservation to the system. Actor defines which type of reservation is needed. Actor enters the required customer information. As the result of this function customer information is stored, a reservation is made and in case of a room or restaurant reservation a confirmation payment slip is printed to the customer.

Confirm and Cancel Reservation
This function allows actor to either confirm an existing reservation or cancel it, respectively. Actor retrieves reservation data and either confirms or cancels it.

3.3.2 Administrative Services

Create Instance
This function allows the actor to create new instances of rooms/tables/table charts in to the system. Actor enters the required data for an instance and confirms that the data is correct.

Show Instance
This function retrieves the data of an instance from the system and displays it to the actor.

Update Instance

This function allows the actor to update the data of an instance after retrieving the corresponding data with the 'Show Instance' – service. For example if a room is renovated so that it gets a shower and a toilet.

Delete Instance

This function allows the user to delete an instance from the system database after retrieving the corresponding data with the 'Show Instance' – service.

3.4 Actors

Staff

A hotel staff user can use all the functions of the system without restrictions: creating reservations and instances, modifying them and confirming and canceling reservations.

Customer

A customer is an on-line hotel customer who has access only to create and show reservation services. The customer has a different view on the total reservations, he/she can only see full data of his/her own recent reservation with the reservation number. Otherwise only the current availability situation is shown, so that the customer is able to see if there are free rooms/tables or if the restaurant is available for reservation on a certain time-line.

4 Use Cases

This part includes the Use Case Descriptions and data required per use case display.

4.1 Use Case Descriptions – Reservation Services

4.1.1 Create Room Reservation

Use case:	Create Room Reservation
Actor:	Customer/Staff
Precondition:	A customer needs to make a reservation for a hotel room or several rooms, depending on the number of people on the stay.
Goal:	Reservation is stored in the system waiting to be confirmed and is given a unique ID.

1. The actor defines that he wants to enter room reservation information into the system.

2. The system displays a form:
    a. the following data fields to be filled in: customer name, phone, email, number of people, number of children, age(s) of children and more information.

    b. the following data to be chosen from: type of room desired, desired view, time of reservation, time of arrival.

3. The actor enters the reservation information into the system: customer name, phone, email (optional), number of people, number of children (optional), age(s) of children (required only if number of children is entered) and selects type of room desired (optional), desired view (optional), time of reservation, time of arrival.

4. The actor defines that he wants to save the reservation information into the system.

5. The system displays a message stating that the data has been successfully saved and gives a unique reservation ID and displays an invoice for the reservation fee and states that the reservation will be confirmed once the invoice is paid.

4.1.2 Show Room Reservation

Use case:	Show Room Reservation
Actor:	Customer/Staff
Precondition:	A customer has made a reservation for a hotel room or several rooms. Reservation ID is known.
Goal:	Reservation data is shown.

1. The actor defines that he wants to see a reservation's data from the system.

2. The system displays a form:
    a. the following data fields to be filled in: reservation ID.

3. The actor enters the reservation ID into the system.

4. The actor defines that he wants to find the reservation with the given ID information from the system.

5. The system displays:
    a. Reservation found with ID. The system displays the reservation information.
    b. Reservation not found. System displays a message that the reservation was not found.

4.1.3 Create Restaurant Reservation

Use case:	Create Restaurant Reservation
Actor:	Customer/Staff
Precondition:	A customer needs to make a reservation for the entire restaurant for a special occasion.
Goal:	Reservation is stored in the system waiting to be confirmed and is given a unique ID. All tables are

automatically reserved for that time the restaurant is reserved.

1. The actor defines that he wants to enter restaurant reservation information into the system.

2. The system displays a form:
   a. the following data fields to be filled in: customer name, phone, email, number of people, more information.

   b. the following data to be chosen from: time of reservation.

3. The actor enters the reservation information into the system: customer name, phone, email (optional), number of people and selects time of reservation.

4. The actor defines that he wants to save the reservation information into the system.

5. The system displays a message stating that the data has been successfully saved and gives a unique reservation ID and displays a message that the reservation will be processed and confirmed by the staff.

4.1.4 Show Restaurant Reservation

Use case:	Show Restaurant Reservation
Actor:	Customer/Staff
Precondition:	A customer has made a reservation for the restaurant. Reservation ID is known.
Goal:	Reservation data is shown.

1. The actor defines that he wants to see a reservation's data from the system.

2. The system displays a form:
    a. the following data fields to be filled in: reservation ID.

3. The actor enters the reservation ID into the system.

4. The actor defines that he wants to find the reservation with the given ID information from the system.

5. The system displays:
    a. Reservation found with ID. The system displays the reservation information.
    b. Reservation not found. System displays a message that the reservation was not found.

4.1.5 Create Table Reservation

Use case:	Create Table Reservation
Actor:	Customer/Staff
Precondition:	A customer needs to make a reservation for a table or several tables in the hotel restaurant on a public serving.
Goal:	Reservation is stored in the system waiting to be confirmed and is given a unique ID.

1. The actor defines that he wants to enter table reservation information into the system.

2. The system displays a form:
    a. the following data fields to be filled in: customer name, phone, email, number of people, more information.

    b. the following data to be chosen from: time of reservation.

3. The actor enters the reservation information into the system: customer name, phone, email (optional), number of people and selects time of reservation.

4. The actor defines that he wants to save the reservation information into the system.

5. The system displays a message stating that the data has been successfully saved and gives a unique reservation ID and displays a message that the reservation will be processed and confirmed by the staff.

4.1.6 Show Table Reservation

Use case:	Show Table Reservation
Actor:	Customer/Staff
Precondition:	A customer has made a reservation for a table/tables. Reservation ID is known.
Goal:	Reservation data is shown.

1. The actor defines that he wants to see a reservation's data from the system.

2. The system displays a form:
    a. the following data fields to be filled in: reservation ID.

3. The actor enters the reservation ID into the system.

4. The actor defines that he wants to find the reservation with the given ID information from the system.

5. The system displays:
    a. Reservation found with ID. The system displays the reservation information.

b. Reservation not found. System displays a message that the reservation was not found.

4.1.7 Confirm Reservation

Use case:	Confirm Reservation
Actor:	Staff
Precondition:	A customer has made a reservation for a table/tables, a room/rooms or the restaurant. Reservation ID is known.
Goal:	Reservation data is shown with the use of 'Show X Reservation' – service. Reservation is confirmed

1. Use case: 'Show X Reservation'. (Replace X with the type of reservation that is to be confirmed.

2. The actor defines that he wants to confirm the reservation.

3. The system displays a list of the available and suitable rooms OR tables (depending on the type of reservation) during the reservation time line.

4. The actor selects the room(s) OR table(s) for the reservation.

5. The actor defines that he wants to confirm the reservation.

6. The system displays:
   a. Reservation confirmed.
   b. Reservation already confirmed.

4.1.8 Cancel Reservation

Use case:	Cancel Reservation
Actor:	Staff

Precondition:	A customer has made a reservation for a table/tables, a room/rooms or the restaurant. Reservation ID is known.
Goal:	Reservation data is shown with the use of 'Show X Reservation' – service. Reservation is canceled.

1. Use case: 'Show X Reservation'. (Replace X with the type of reservation that is to be confirmed.

2. The actor defines that he wants to cancel the reservation.

3. The system displays: Reservation canceled.

4.2 Use Case Descriptions – Instance Services

4.2.1 Create Room

Use case:	Create Room
Actor:	Staff
Precondition:	A new room instance needs to be entered into the system.
Goal:	Room data is saved into the system.

1. The actor defines that he wants to create a new room into the system.

2. The system displays a form:
   a. the following data fields to be filled in: room number, maximum number of people staying in the room with extra beds included, more information

   b. the following data to be chosen from: type of room, type of view, type of bathroom

3. The actor enters the room information into the system: room number, maximum number of people staying in the room with extra beds included, more information (optional) and selects type of room, type of view, type of bathroom.

4. The actor defines that he wants to save the room information into the system.

5. The system displays a message:
    a. the data has been successfully saved.
    b. the data was not saved, room number already in use.

4.2.2 Show Room

Use case:	Show Room
Actor:	Staff
Precondition:	Room data has been saved into system. Room number is known.
Goal:	Room data is shown.

1. The actor defines that he wants to see a room's data from the system.

2. The system displays a form:
    a. the following data fields to be filled in: room number.

3. The actor enters the room number into the system.

4. The actor defines that he wants to find the room with the given room number information from the system.

5. The system displays:
    a. Room found with ID. The system displays the room information.

b. Room not found. System displays a message that the room was not found.

4.2.3 Update Room

Use case:	Update Room
Actor:	Staff
Precondition:	Room needs to be updated in the system. Room data is shown with the 'Show Room' – service.
Goal:	Room data is updated.

1. Use case: 'Show Room'.

2. The actor defines that he wants to update the room data.

3. The system displays a form with the existing room data:
   a. the following data fields to be filled in: room number, maximum number of people staying in the room with extra beds included, more information

   b. the following data to be chosen from: type of room, type of view, type of bathroom

4. The actor makes the necessary changes to the room's data.

5. The actor defines that he wants to save the updates into the system.

6. The system displays: Updated data saved.

4.2.4 Delete Room

Use case:	Delete Room
Actor:	Staff

Precondition:	Room needs to be removed from system. Room data is shown with the 'Show Room' – service.
Goal:	Room is deleted.

1. Use case: 'Show Room'.

2. The actor defines that he wants to delete the room.

3. The system displays: Room deleted.

4.2.5 Create Table

Use case:	Create Table
Actor:	Staff
Precondition:	A new table instance needs to be entered into the system.
Goal:	Table data is saved into the system and the table instance is given a unique ID.

1. The actor defines that he wants to create a new table into the system.

2. The system displays a form:
   a. the following data fields to be filled in: table number, maximum number of people sitting in the table, more information.

   b. the following data to be chosen from: type of table.

3. The actor enters the table information into the system: table number, maximum number of people sitting in the table, more information (optional) and selects type of table.

4. The actor defines that he wants to save the table information into the system.

5. The system displays a message:
    a. the data has been successfully saved.
    b. the data was not saved, table number already in use.

4.2.6 Show Table

Use case:	Show Table
Actor:	Staff
Precondition:	Table data has been saved into system. Table number is known.
Goal:	Table data is shown.

1. The actor defines that he wants to see a table's data from the system.

2. The system displays a form:
    a. the following data fields to be filled in: table number.

3. The actor enters the table number into the system.

4. The actor defines that he wants to find the table with the given table number information from the system.

5. The system displays:
    a. Table found with ID. The system displays the table information.
    b. Table not found. System displays a message that the table was not found.

4.2.7 Delete Table

Use case:	Delete Table
Actor:	Staff

Precondition:	Table needs to be removed from system. Table data is shown with the 'Show Table' – service.
Goal:	Table is deleted.

1. Use case: 'Show Table'.

2. The actor defines that he wants to delete the table.

3. The system displays: Table deleted.

4.2.8 Create Table Chart

Use case:	Create Table Chart
Actor:	Staff
Precondition:	A new table chart instance needs to be entered into the system.
Goal:	Table Chart data is saved into the system.

1. The actor defines that he wants to create a new table chart into the system.

2. The system displays a form:
   a. the following data fields to be filled in: number of people, number of tables, more information

   b. the following data to be chosen from: types of tables

3. The actor enters the table chart information into the system: number of people, number of tables, more information (optional) and selects types of tables.

4. The actor defines that he wants to save the table chart information into the system.

5. The system displays a message: the data has been successfully saved.

6. The system displays an image with the table chart.

4.2.9 Show Table Chart

Use case:	Show Table Chart
Actor:	Staff
Precondition:	Table Chart data has been saved into system. Table Chart ID is known.
Goal:	Table Chart data is shown.

1. The actor defines that he wants to see a table chart's data from the system.

2. The system displays a form:
    a. the following data fields to be filled in: table chart ID.

3. The actor enters the table chart ID into the system.

4. The actor defines that he wants to find the table chart with the given table chart ID from the system.

5. The system displays:
    a. Table Chart found with ID. The system displays the table chart information. The system displays the table chart image.
    b. Table Chart not found. System displays a message that the table chart was not found.

4.2.10 Update Table Chart

Use case:	Update Table Chart
Actor:	Staff

Precondition:	Table Chart needs to be updated in the system. Table Chart data is shown with the use of 'Show Table Chart' – service.
Goal:	Table Chart data is updated.

1. Use case: 'Show Table Chart'.

2. The actor defines that he wants to update the table chart data.

3. The system displays a form:
   a. the following data fields to be filled in: number of people, number of tables, more information
   b. the following data to be chosen from: types of tables

4. The actor makes the necessary changes to the room's data.

5. The actor defines that he wants to save the updates into the system.

6. The system displays: Updated data saved.

7. The system displays an image with the updated table chart.

4.2.11 Delete Table Chart

Use case:	Delete Table Chart
Actor:	Staff
Precondition:	Table Chart needs to be removed from system. Table Chart data is shown with the 'Show Table Chart' – service.
Goal:	Table Chart is deleted.

1. Use case: 'Show Table Chart'.

2. The actor defines that he wants to delete the table chart.

3. The system displays: Table Chart deleted.

4.3 Data Requirements for Displays – Reservation Services

4.3.1 Data Requirements for Create Room Reservation

Actor enters:

- customer name
- phone
- email
- number of people
- number of children
- ages of children
- more information

Actor selects:

- type of room desired
- desired view
- time of reservation
- time of arrival

System shows:

- confirmation of creation message
- reservation ID and details
- reservation fee invoice
- information message regarding the confirmation of the reservation once the invoice is paid

4.3.2 Data Requirements for Show Room Reservation

Actor enters:

- reservation ID

System shows:

- reservation data OR

- information message regarding the reservation was not found.

4.3.3 Data Requirements for Create Restaurant Reservation

Actor enters:

- customer name
- phone
- email
- number of people
- more information

Actor selects:

- time of reservation

System shows:

- confirmation of creation message
- reservation ID and details
- information message regarding the confirmation of the reservation once it is processed by the staff

4.3.4 Data Requirements for Show Restaurant Reservation

Actor enters:

- reservation ID

System shows:

- reservation data OR
- information message regarding the reservation was not found.

4.3.5 Data Requirements for Create Table Reservation

Actor enters:

- customer name
- phone
- email
- number of people

- more information

Actor selects:

- time of reservation

System shows:

- confirmation of creation message
- reservation ID and details
- information message regarding the confirmation of the reservation once it is processed by the staff

4.3.6 Data Requirements for Show Table Reservation

Actor enters:

- reservation ID

System shows:

- reservation data OR
- information message regarding the reservation was not found.

4.3.7 Data Requirements for Confirm Reservation

System shows:

- information message regarding the reservation was confirmed OR
- information message regarding the reservation was already confirmed.

4.3.8 Data Requirements for Cancel Reservation

System shows:

- information message regarding the reservation was canceled.

4.4 Data Requirements for Displays – Instance Services

4.4.1 Data Requirements for Create Room

Actor enters:

- room number
- maximum number of people staying in the room with extra beds
- more information

Actor selects:

- type of room
- type of view
- type of bathroom

System shows:

- confirmation of creation message OR
- error message stating that the room number is already in use.

4.4.2 Data Requirements for Show Room

Actor enters:

- room number

System shows:

- room data OR
- information message regarding the room was not found.

4.4.3 Data Requirements for Update Room

Actor enters:

- room number
- maximum number of people staying in the room with extra beds
- more information

Actor selects:

- type of room
- type of view
- type of bathroom

System shows:

- confirmation of update message

4.4.4 Data Requirements for Delete Room

System shows:

- information message regarding the room was deleted.

4.4.5 Data Requirements for Create Table

Actor enters:

- table number
- maximum number of people sitting in the table
- more information

Actor selects:

- type of table

System shows:

- confirmation of creation message OR
- error message stating that the table number is already in use.

4.4.6 Data Requirements for Show Table

Actor enters:

- table number

System shows:

- table data OR
- information message regarding the table was not found.

4.4.7 Data Requirements for Delete Table

System shows:

- information message regarding the table was deleted.

4.4.8 Data Requirements for Create Table Chart

Actor enters:

- number of people

- number of tables
- more information

Actor selects:

- types of tables

System shows:

- confirmation of creation message
- table chart image
- unique table chart ID

4.4.9 Data Requirements for Show Table Chart

Actor enters:

- table chart ID

System shows:

- table chart data
- table chart image OR
- information message regarding the table chart was not found.

4.4.10 Data Requirements for Update Table Chart

Actor enters:

- number of people
- number of tables
- more information

Actor selects:

- types of tables

System shows:

- confirmation of update message
- updated table chart image

4.4.11 Data Requirements for Delete Table Chart

System shows:

- information message regarding the table chart was deleted.

5 Class Model

This part consists of the Class diagram, Class descriptions and State Chart diagrams showing the life cycles of Reservation and Instance objects.

5.1 Class Diagram

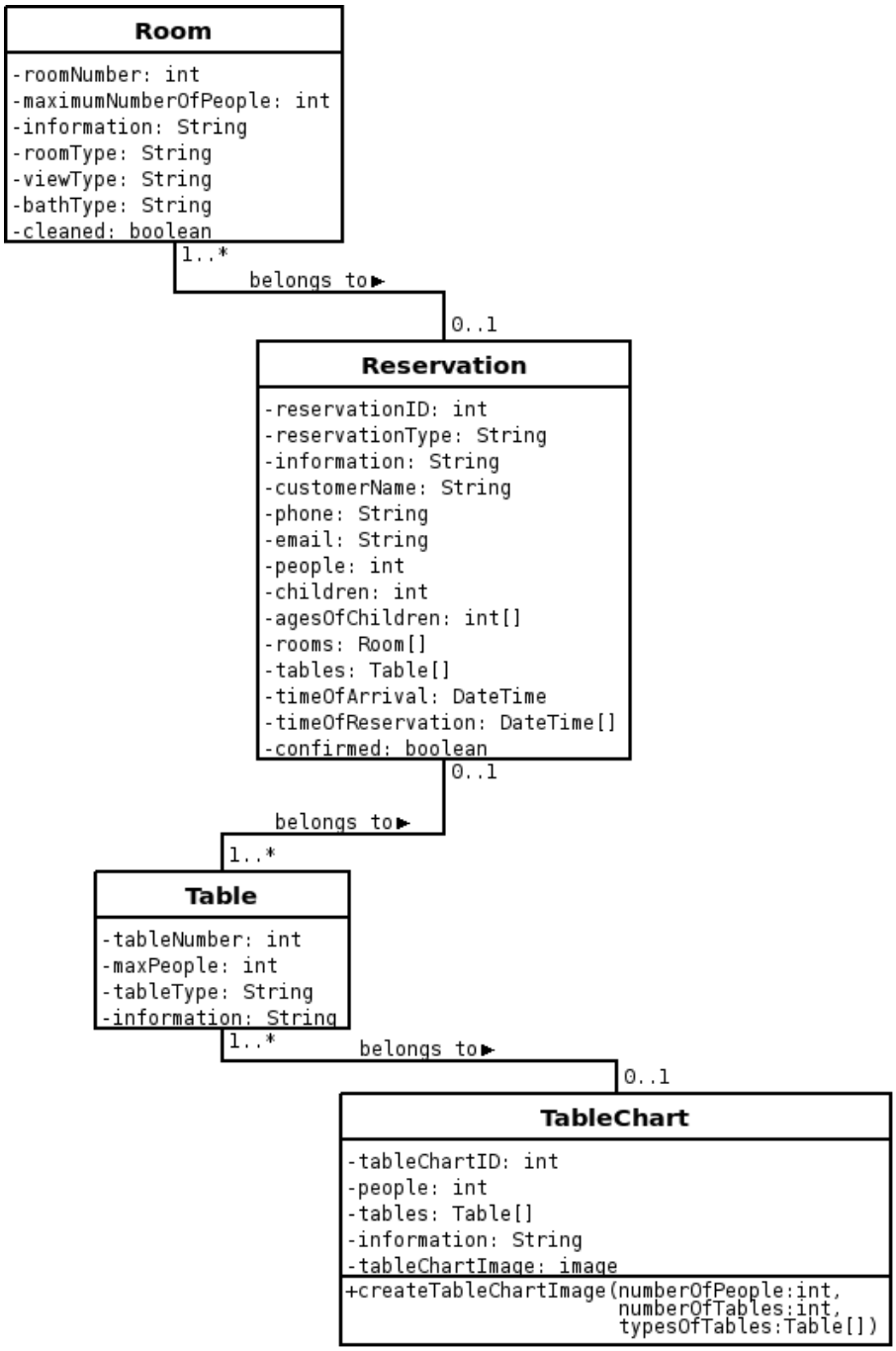

Figure 2. Class Diagram of the System

5.2 Class Descriptions

This part describes the classes in detail.

5.2.1 Reservation

Class	Reservation
Definition	Reservation is made to hold a room, table or the hotel restaurant for someone for a certain time limit.
Attributes	***reservation ID*** – numeric, 6 digits, no decimals – Mandatory System determines reservation ID for new reservations from a continuous set of numbers. Identifies reservation uniquely. **reservationType** – alphabetic, 11 characters – Mandatory Tells the type of the reservation: room, table or restaurant. Restaurant reservation reserves all tables. **information** – alphabetic, 200 characters – Optional Other notations user can make to clarify the reservation. ***customerName*** – alphabetic, 30 characters – Mandatory The name of the hotel visitor who wishes to make the reservation. **phone** – alphabetic, 12 characters – Mandatory Contact number of the customer, mobile or fixed phone. **email** – alphabetic, 30 characters – Optional Contact email address of the customer. **people** – numeric, 3 digits, no decimals – Mandatory

	Number of people the reservation concerns. **children** – numeric, 3 digits, no decimals – Optional Number of children the reservation concerns. **agesOfChildren** – numeric, 2 digits, no decimals – Mandatory only if children is not null. **rooms** – array of room objects – Mandatory only if reservationType is room. Tells the number of unique rooms in the reservation. **tables** – array of table objects – Mandatory only if reservationType is table or restaurant. In case of restaurant reservation, all tables are reserved. **timeOfArrival** – date and time – Mandatory The time when the reservation starts, in case of a room reservation it can be a time line e.g. from 12pm to 4pm. **timeOfReservation** – date and time, 2 values, Mandatory only is reservationType is room. The time of the stay in the hotel, meaning how long the room/rooms, need to be reserved by the customer. **confirmed** – boolean, true/false, Mandatory Tells if the reservation has been confirmed by the staff or not. Initially false.
Associations	“has” association with zero or more Room “has” association with zero or more Table
Responsibility	“knows” its Room(s) “knows” its Table(s)
Operations	create reservation() confirm reservation() cancel reservation()

5.2.2 Room

Class	Room
Definition	Room is an instance in the hotel where visitors can spend time and can reserve before hand. It can also mean a cabin, which is also used in the same manner.
Attributes	***roomNumber*** – numeric, 3 digits, no decimals – Mandatory Identifies Room uniquely. **maximumNumberOfPeople** – numeric, 2 digits – Mandatory Tells how many people can stay in the room at a time. **information** – alphabetic, 200 characters – Optional Other notations user can make to clarify the Room. **roomType** – alphabetic, 15 characters – Mandatory Tells the type of the room: single, two-person, five-person or a cabin. **viewType** – alphabetic, 15 characters – Mandatory Tells the type of the view from the room: whether it is to the street or to the sea. **bathType** – alphabetic, 15 characters – Optional Tells the type of the bathroom in the room: whether it is a simple toilet or with a shower. **cleaned** – boolean, true/false, Mandatory Tells if the room has been cleaned by the staff or not. Initially true.
Associations	“belongs to” association with zero or more Reservation
Responsibility	“knows” its Reservation(s)
Operations	create room() update room() show room()

delete room()

5.2.3 Table

Class	Table
Definition	Table is an instance in the hotel restaurant where visitors can enjoy a meal or special occasion and can reserve before hand.
Attributes	***tableNumber*** – numeric, 3 digits, no decimals – Mandatory Identifies Table uniquely. **maxPeople** – numeric, 2 digits – Mandatory Tells how many people can sit in the table at a time. **tableType** – alphabetic, 15 characters – Mandatory Tells the type of the table: round, square or rectangular. **information** – alphabetic, 200 characters – Optional Other notations user can make to clarify the Table.
Associations	“belongs to” association with zero or more Reservation “belongs to” association with zero or more TableChart
Responsibility	“knows” its Reservation(s) “knows” its TableChart(s)
Operations	create table() show table() delete table()

5.2.4 TableChart

Class	TableChart
Definition	TableChart is an instance in the hotel restaurant where visitors can enjoy a meal or special occasion and can reserve before hand.
Attributes	***tableChartID*** – numeric, 3 digits, no decimals – Mandatory System determines reservation ID for new reserva-

	tions from a continuous set of numbers. Identifies TableChart uniquely. **people** – numeric, 3 digits – Mandatory Tells the total number of people in the table reservations that are used to make the table chart. **tables** – array of table objects – Mandatory The table instances used by this table chart. **information** – alphabetic, 200 characters – Optional Other notations user can make to clarify the TableChart. **tableChartImage** – image An image of the table chart generated by the system.
Associations	“has” association with zero or more Table
Responsibility	“knows” its Table(s)
Operations	create tableChart() show tableChart() update tableChart() delete tableChart() create tableChartImage

5.3 Objects’ Life Cycles

The following state chart diagrams show Room and Reservation’s objects’ life cycles.

5.3.1 Room Object

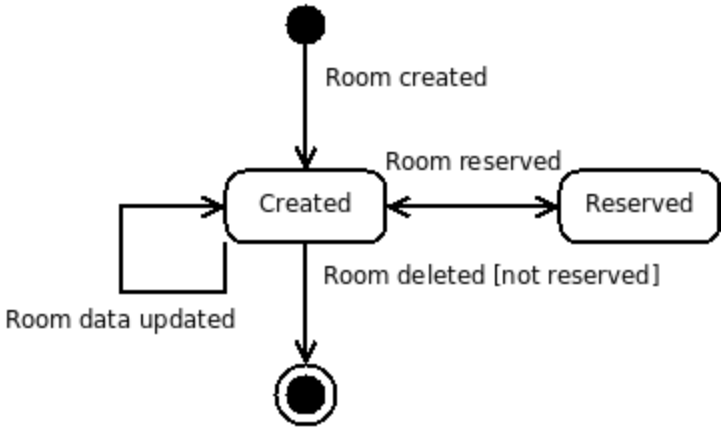

Figure 3. State Chart Diagram for Room Object

5.3.2 Reservation Object

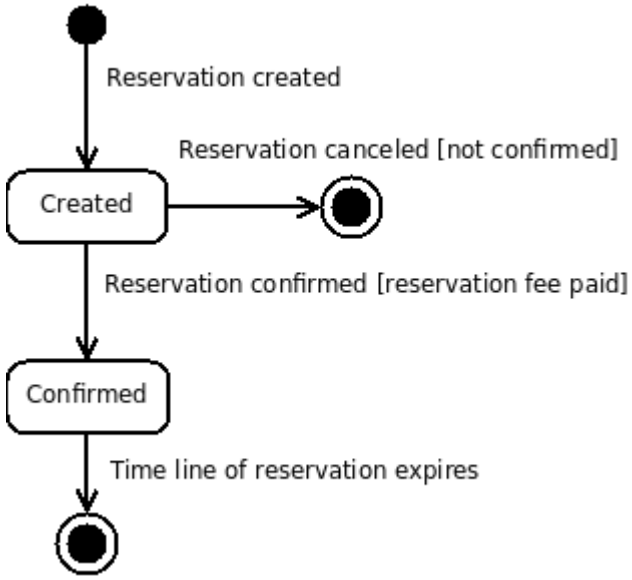

Figure 4. State Chart Diagram for Reservation Object

6 Summary of Data Use

Classes **Use cases**	Reservation	Room	Table	TableChart
Create Room Reservation	C	R		
Show Room Reservation	R	R		
Create Table Reservation	C		R	
Show Table Reservation	R		R	
Create Restau-rant Reservation	C		R	
Show Restaurant Reservation	R		R	
Confirm Reserva-tion	U	R*	R*	
Cancel Reserva-tion	U	R*	R*	
Create Room		C		
Show Room		R		
Update Room		U		
Delete Room		U		
Create Table			C	
Show Table			R	
Delete Table			U	
Create Table Chart				C
Show Table Chart				R
Update Table Chart				U
Delete Table Chart				U

C- Create; R- Read; U- Update* depending on the type of reservation.

Table 1. Summary of data use

7 Other Requirements

7.1 Rights of access

The services designed for customer users are meant to be public. The administrative 'staff' services are controlled by user name and password login.

7.2 Minimizing manual input

The basic principle is to maximize the use of once stored data and avoiding redundant information and multiple inputting.

7.3 System interfaces

System interfaces must be kept clearly defined and well managed.

7.4 Error handling and error messages

System should include efficient error preventing procedures and error messaging should be clear and informative.

8 Security

Actor **Use case**	Customer	Staff
Create Room Reservation	x	x
Show Room Reservation	x	x
Create Table Reservation	x	x
Show Table Reservation	x	x
Create Restaurant Reservation	x	x
Show Restaurant Reservation	X	x
Confirm Reservation		x
Cancel Reservation		x
Create Room		x
Show Room		x
Update Room		x
Delete Room		x
Create Table		x
Show Table		x
Delete Table		x
Create Table Chart		x
Show Table Chart		x
Update Table Chart		x
Delete Table Chart		x

Table 2. Security table showing which use cases are available for which actor type.

9 Test Cases

9.1 Use Case Instance: Create Room Reservation

A new Room Reservation is created in the system. All optional fields are entered except 'more information'.

9.1.1 Scenario: Create Room Reservation

Use case:	Create Room Reservation
Actor:	Customer/Staff
Precondition:	A customer needs to make a reservation for a hotel room or several rooms, depending on the number of people on the stay.
Goal:	Reservation is stored in the system waiting to be confirmed and is given a unique ID.

1. The actor defines that he wants to enter room reservation information into the system.

2. The system displays a form:
   a. the following data fields to be filled in: customer name, phone, email, number of people, number of children, age(s) of children and more information.

   b. the following data to be chosen from: type of room desired, desired view, time of reservation, time of arrival.

3. The actor enters the reservation information into the system: customer name, phone, email (optional), number of people, number of children (optional), age(s) of children (required only if number of children is entered) and selects type of room desired (optional), desired view (optional), time of reservation, time of arrival.

4. The actor defines that he wants to save the reservation information into the system.

5. The system displays a message stating that the data has been successfully saved and gives a unique reservation ID and displays an invoice for the reservation fee and states that the reservation will be confirmed once the invoice is paid.

9.1.2 Collaboration Diagram: Create Room Reservation

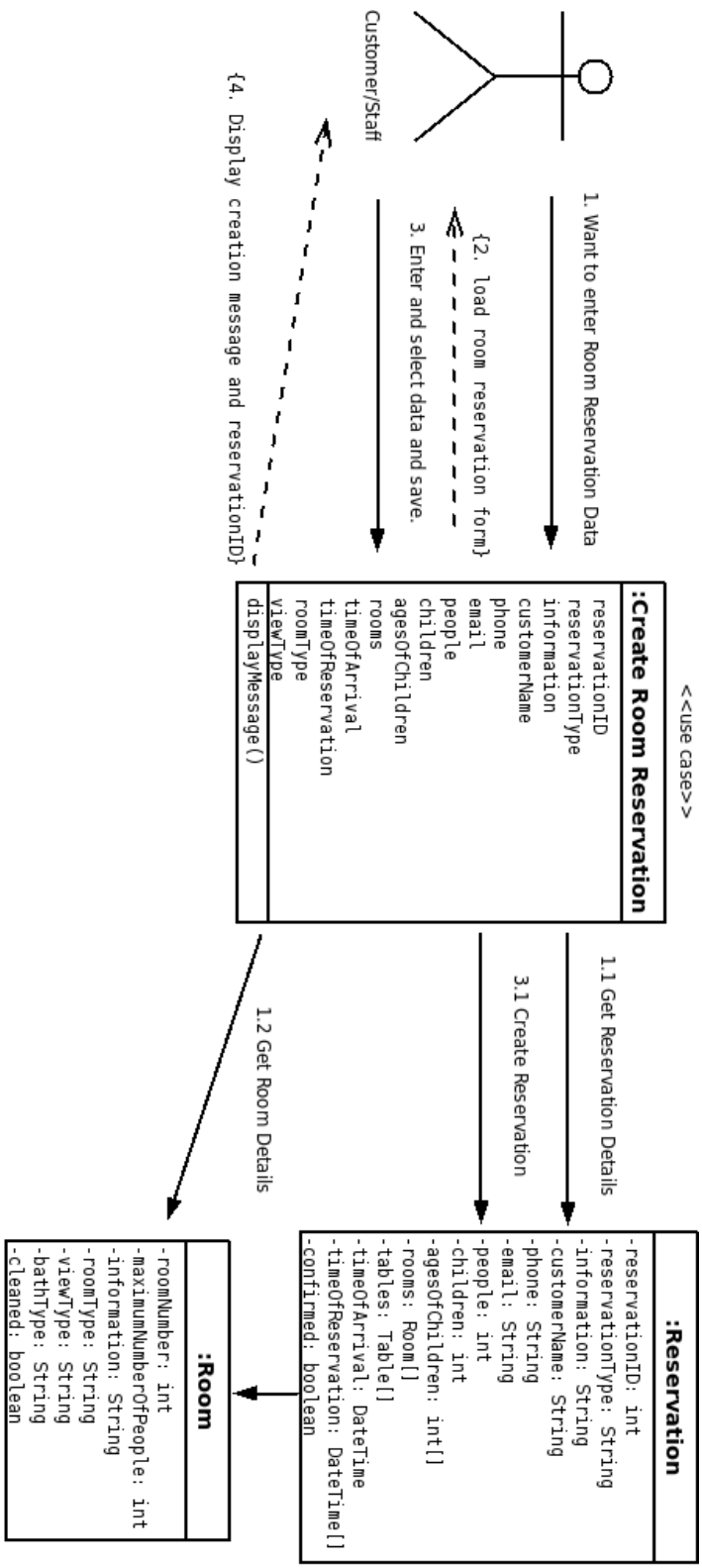

Figure 5. Collaboration Diagram of the Create Room Reservation Use case

10 Design and Database

This part of the document focuses on the design and database model of the system. Possible user interface solutions are presented as well as the relational model of the database.

11 User Interface Description

This part of the document describes the user interface of the system, which will be a graphical web interface. To enrich the interface, ITMill toolkit was planned to be used, but as ITMill is now known as Vaadin, that should be used. Basically it is a framework to enable creation of HTML-forms with rich capabilities (such as tooltips with HTML-formatting) using only Java-language in the programming.

11.1 Example Form Items

Here are sample images of rich elements done with the Vaadin Framework.

Figure 6. Tooltip with rich formatting (Vaadin Sampler)

Figure 7. Date selection for reservations (Vaadin Sampler)

11.2 Create Reservation Form Page

Here is a draft of the reservation form page for the system. This has been made using pure HTML-code.

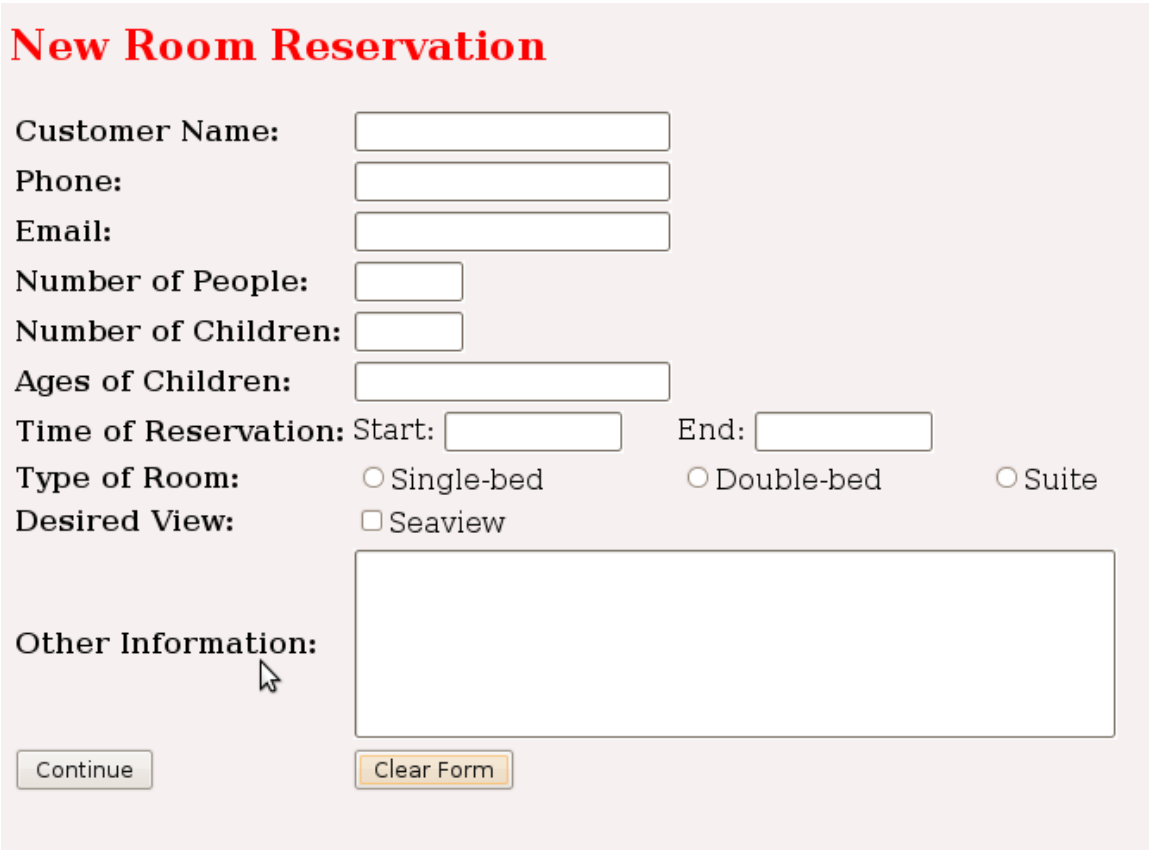

Figure 8. Example Reservation form.

12 Database Overview

This part contains the plan of the system database, including the relational model and relational formulas. It also contains the scripts for creating the database and tables and insert scripts for test data.

The database of the system will be used to contain data of the hotel's customer's reservations and the hotel contents, such as rooms (HotelRoom) and restaurant tables (RestaurantTable). Also the data of table charts (RestaurantTableChart) will be stored. There are helper tables between the relation of Reservation and HotelRoom tables; Reservation and RestaurantTables; RestaurantTable and RestaurantTableChart tables. These are used to avoid many-to-many relations as one hotel room for instance can belong to several reservations and on the other hand one reservation (of room type) can contain several rooms.

12.1 Relational Model

In this part we have the relational model diagram with the primary keys (PK) and foreign keys (FK) of each table. The model is based on the class diagram in the definition part.

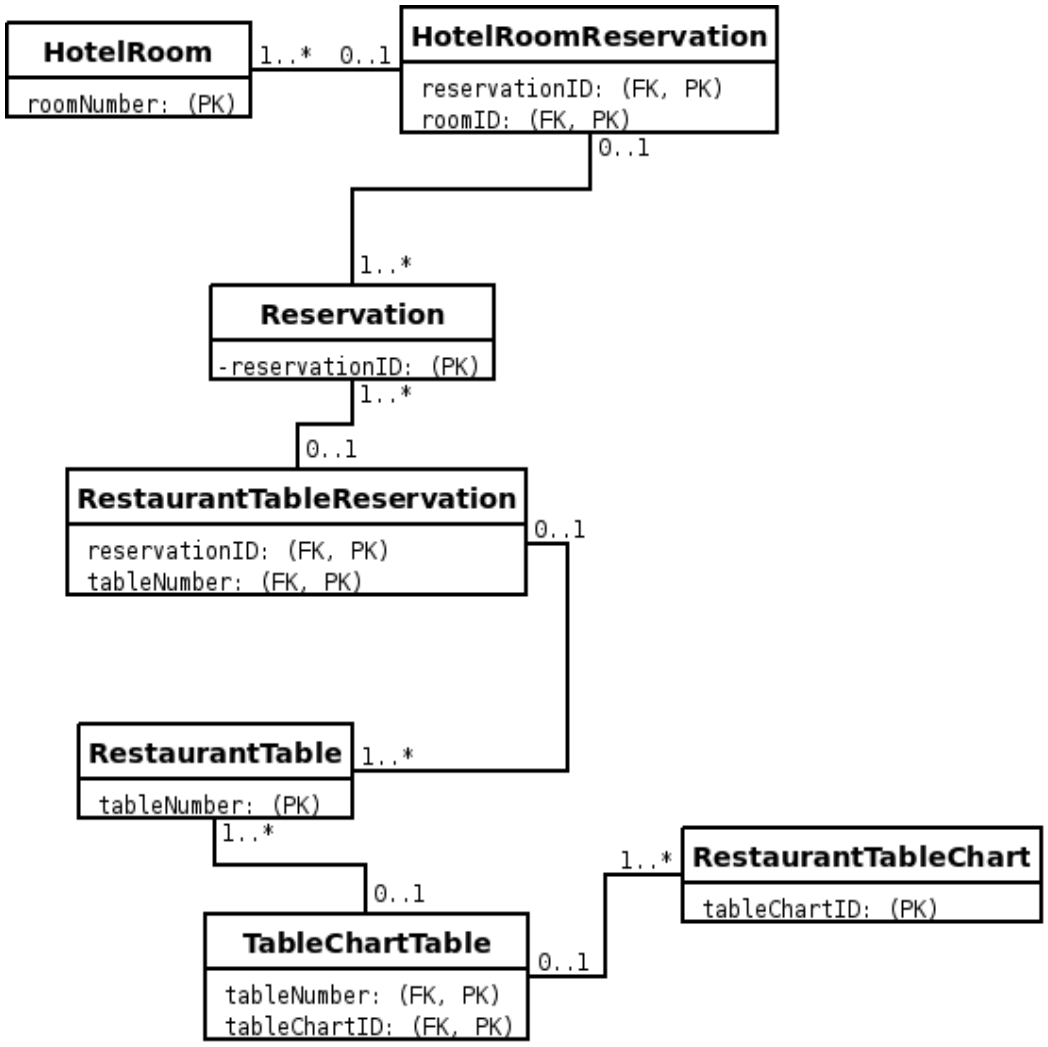

Figure 9. Relational Model of the Database

12.2 Create and Insert Script

The script to create the database and tables and insert test data into them is included in the project data files as an SQL-file.

13 Testing Plan

This part describes the test case planning for the implementation and testing period. The idea is to use the Use Cases specified in the system definition documentation as basis for each JUnit-test case. The main focus of the JUnit tests is in database connectivity and query checking. Following is an example test case written using JUnit for the create room test case.

```
package testing;

import thesis.vpukkila.hotelsystem.control.RoomController;
import junit.framework.TestCase;

public class AddRoomTest extends TestCase {

          RoomController rc = new RoomController();

          public void testCreateRoom() {
                    assertTrue("Room not saved", rc.CreateRoom(2, "",
"Single", "Sea", "WC-shower") == "Room saved in DB");
          }

}
```

14 Implementation Constraints

There are certain rules and constraints that must be followed when implementing the use cases that have not been implemented in the database design. These constraints are listed here.

14.1 Reservation Times

A reservation can not start earlier than the current date and time. The end date of a room reservation is at least one day after the start date. Table reservations do not require an end date as they are meant to reserve for one day only.

14.2 Confirming Reservations

When confirming a room reservation, the system must check the available rooms for that day (or days) and create rows in the HotelRoomReservation-table in the database to specify which rooms are used by that reservation.

When confirming a table reservation, the system must check the available tables for that day and create rows in the RestaurantTableReservation-table in the database to specify which restaurant tables are used by that reservation.

The before mentioned checking should be done using the mentioned tables and checking other reservation data.

15 Implementation of the System

This part explains the solutions used in the implementation of the defined and designed Hotel System in this thesis. The controller, manager and data accessing classes are described in detail. Also other uses of the Spring MVC framework are explained.

Some of the code is commented with the functionality of the classes, methods and attributes. Those comments can be used as a reference with this documentation.

16 Setting Up and Running the Application

The application is designed and implemented on the Java EE platform with a MySQL database in the back-end and Spring MVC (model-view-controller) framework managing and structuring the application functionality. To set up the project, a MySQL database, a Tomcat 5.5 application server and Java 1.5 runtime environment are required. The applica-

tion specific properties for the database and application server connections are included in the project data.

17 Implemented Services

The main services in the project's scope have been implemented. 'Create Reservation' and 'Confirm Reservation' are ready for use and can be easily extended and modified if necessary. Other services in the project scope, such as the room or restaurant table management in the administration services, are not implemented, but the database contains tables for their data, as rooms are needed for the basic reservations.

Services that were not defined that well, 'List Reservations', 'Display Reservation Details' and 'List Available Rooms' are implemented as well as they are included in the 'Confirm Reservation' service. The 'List Reservations' can be used to list all confirmed reservations and selecting one from the list uses the 'Display Reservation Details' service, which in turn uses 'List Available Rooms'. This can be seen in the screenshots below.

17.1 Create Reservation

This service enables the user to create a room reservation for the hotel for a specified time. A validator is used to check the contents filled out by the user in the form. These follow the business logic constraints specified in the design documentation. Some additional rules would have been possible to implement, such as checking the time-line given by the user if there are available rooms in the hotel, in order to avoid reservations to a 'no vacancy' hotel.

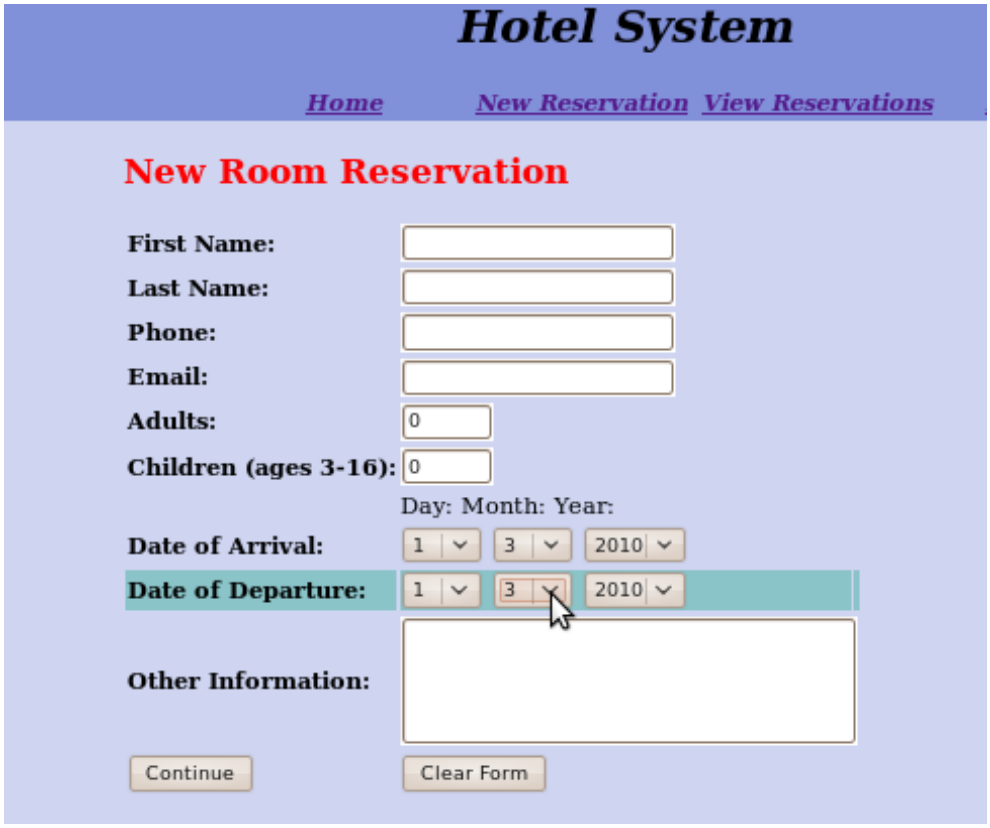

Figure 10. Implemented form for a room reservation.

In the screenshot above, the form for a new reservation is displayed. Once the correct data is entered and 'Continue' is pressed, the user is presented with the following view with instructions how to get their reservation confirmed by the hotel.

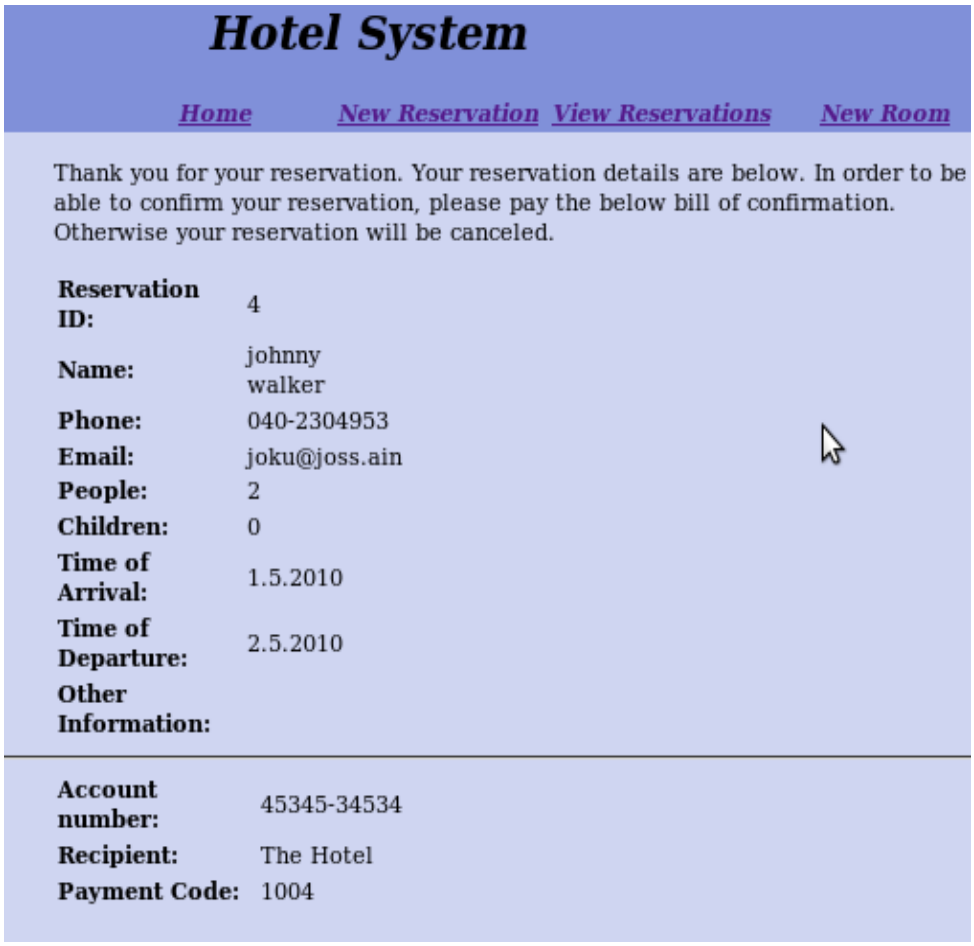

Figure 11. View for customer to pay the confirmation fee.

17.2 Confirm Reservation

This service enables the user to confirm a reservation and select the desired available room for the reservation. The validation rules of the service are not strict enough as there are loop holes and bugs remaining to be fixed. Nevertheless, the service is usable. Following is a view of a listing of the reservations that haven't been confirmed.

Figure 12. Implemented view for listing reservations.

From this view, the user can click on the name of the reserver, which opens a new view with details of that reservation. This view is displayed in the next image.

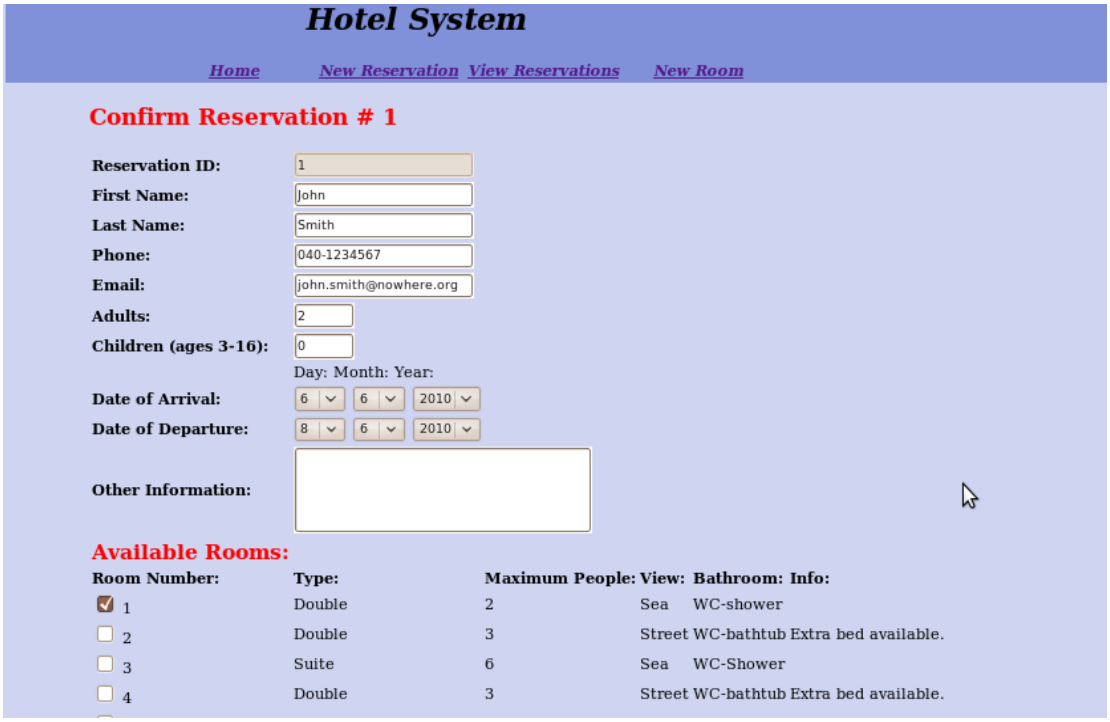

Figure 13. Implemented form and view for confirming a reservation.

In addition to just confirming the reservation by selecting the rooms and pressing confirm, the user can update the data of the reservation if necessary, such as if there are more or less people coming or if the time is changed.

18 Application Structure

This part specifies the different areas of the application and how they are implemented. Also the database and data management is briefly explained.

18.1 User Interfaces and Controller Classes

The user interfaces are created with JSP and Spring MVC framework is used on the forms. A dispatcher servlet has been defined in the web.xml file and it handles all http-requests on the application. The servlet points to specific controllers, mainly the HotelController class when a simple page without a form is requested, and to Form-specific controllers in case a form page is requested. The HotelController class checks the request path for each page and forwards to an appropriate view with a model containing data that might be needed on the page.

One example is the listReservations-view which displays all unconfirmed reservations. The controller gets the data using an instance of the ReservationManager class and puts it to the model which is then referenced on the displayed page and JSTL foreach loop is used to iterate over the collection of Reservation objects in the model to get the required data of each object.

18.1.1 Vaadin / ITMill Toolkit Rich User Interface Framework

The Vaadin framework (previously known as ITMill toolkit) was planned to be used in the project to create rich user interfaces with AJAX-style functionality and look. The framework uses Javascript to create rich elements in user interfaces and it can be programmed with pure Java. The framework was installed and tested in a basic mode, but when a more complex form was created with a Custom Component mode, there were problems running the application. Nothing was shown on the browser screen and numerous Google-searches, project debugs and retries later,

the framework was removed from the project in order to get things done.

18.2 Business Logic and Data Managers

The data required on the pages and data sent from forms to the system is handled by manager classes, ReservationManager and RoomManager. They have specific methods for accessing data from the data access objects, which in turn communicate with the database. The manager classes and the form validator classes handle also most of the responsibility of data integrity. These include such points as the correct form of an email address and reservation dates handling. There are points in the business logic that have not been yet implemented or haven't been tested properly due to lack of time.

18.3 Database and Data Access Objects

The database of the system has been defined in the Design documentation and some small changes were required in the implementation phase. The main objects in the use of the system are Reservation and Room, with the tables in the database Reservation, HotelRoom and HotelRoomReservation. The last table is used when reservations are confirmed and each room needs to be checked if it is available for a reservation in the time required by that reservation.

The complete SQL dump file of the created database with some anonymous test data is included in the project data along with the source codes.

19 Testing

The original plan was to test properly the system and write appropriate test classes for each service but as the time usage of the project was unsuccessful, this was the unfortunate part to be diminished. There are

a couple of unit tests in the package thesis.vpukkila.hotelsystem.testing, but their functionality and importance is minimal.

The usability and availability of the implemented services has been tested the whole time they have been developed from the user's point of view. This has been important as there hasn't been enough time to create proper test cases or integration tests. Bugs and defects in the code have been found and fixed with a trial and error method. Once a feature hasn't worked, it has been investigated and repaired if possible.

20 Summary and Report

20.1 Achieved Goals During the Project

When making a comparison on the achievements of the project with the planned scope, there is a great difference. The main reason for this was that project was planned to be large, so that there would be enough to do. This can be seen in the definition documentation and the database model as they are ready for a full system with all the services in the project scope.

The implemented services in the system are 'Create Reservation' and 'Confirm Reservation'. Administration services such as 'Create Room' were started, but not finished due to lack of time. An estimated 40-60 hours more work on implementation would have probably made a major difference.

20.1.1 Learning Goals

In the end the most important thing in the project was the student's learning which improved especially when working on a Java Spring MVC application. Also the more complex database queries required on the 'Confirm Reservation' service made the student a better data handler.

The system is ready for usage on the hotel side, but the security issue of separating regular users from hotel administration is not implemented, so it shouldn't be used on-line as a public service. This should be possible to fix in the above mentioned suggested time.

20.2 Project Management and Implementation Difficulties

The project was well planned and the scope was suitable, but the plan would have required more work from the student during the fall and winter. Due to external issues with illness, work and family, the time available for the thesis project diminished and so the project completion took more time.

All in all, it took for instance three tries to create user interfaces as forms didn't work properly with Vaadin Framework, then the plain HTML-version that was imported to the Java project didn't submit properly so it needed to be done again from scratch to be able to use the Spring Framework capabilities and then it finally worked. With more time available for the project, the results would have been better and more organized. Now the program code is working but buggy, as time had to be cut from proper testing.

21 Bibliography

Interview with Hotel Manager Maritta Lehtinen, August 31st 2009.

Vaadin Sampler, http://demo.vaadin.com/sampler, date referenced: 3.12.2009.