

**Republic of Yemen**

**University of Science and Technology**

**Faculty of Computers and IT**

**HMS**

**Hotels Management System Online**

**Student’s Name**

Abdulrahman Mohammed Shaiban

Emad Abdulkareem Alareefi

**Supervisor**

Dr. Asma’a Al-Sharjabi

# ACKNOWLEDGMENT

First, all praise to Allah, the most Gracious and most Merciful, for the many blessings He bestowed upon us. All thanks are to Allah who gave us the ability and patience throughout our studies for completing the degree program.

Completion of this project would not have been possible without the efforts of our advisor and mentor – Dr. Asma Alshargabi, words cannot describe our gratitude for her guidance and support with this project.

We are thankful for everyone who has helped with the research behind this project. Finally, nobody has been more important to us in the pursuit of this project than the member of our families. We would like to thank our parents and siblings, whose love and guidance are with us in whatever we pursue. They are the ultimate role models. Most importantly, we wish to thank our doctors, teachers and University.

# 

# Declaration

We hereby certify that this material, which we now submit for assessment on the programme of study leading to the award of Bachelor of Science in (Information Technology, Information System) is entirely our own work, that We have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

**Signed:** Abdulrahman Shaiban, Emad Alareefi

**Registration No:**

**Date:**

# Abstract

In the entire world, the number of hotels has increased. In future, this range will get increased, All these hotels need systems to manage all its functions and processes, In light of this great need for systems, a lot of hotel management systems showed up, but it's not easy to use and it is very expensive.

This project present a hotel management system online this system can be accessed from anywhere; it can manage all the functions in the hotels. It designed by using NodeJS with MongoDB, this system provide easy interfaces for the hotels where it can reserve rooms, add customer data, shows rooms cases, accounts management, services management, purchases management and generate reports. It is easy to use and very cheap.

## Table of Contents

[Declaration ii](#_Toc514845416)

[Abstract ii](#_Toc514845417)

[Table of Contents iii](#_Toc514845418)

[List of figures vi](#_Toc514845419)

[Chapter 1 1](#_Toc514845420)

[Introduction 1](#_Toc514845421)

[1.1 Project Title 2](#_Toc514845422)

[1.2 Introduction 2](#_Toc514845423)

[1.3 Problem Statement 2](#_Toc514845424)

[1.4 Project Objectives 3](#_Toc514845425)

[1.5 Project Scope 3](#_Toc514845426)

[1.6 Project Methodology 4](#_Toc514845427)

[1.7 Tools 4](#_Toc514845428)

[Chapter 2 5](#_Toc514845429)

[Literature review 5](#_Toc514845430)

[2. 1 Introduction 6](#_Toc514845431)

[2. 2 Nazeel Management System 6](#_Toc514845432)

[2.2.1 Login page 6](#_Toc514845433)

[2.2.2 Main page 7](#_Toc514845434)

[2.2.3 Reservations 7](#_Toc514845435)

[2.2.4 Furniture 9](#_Toc514845436)

[2.2.5 Customers 10](#_Toc514845437)

[2.2.6 Services 10](#_Toc514845438)

[2.2.7 Bills 11](#_Toc514845439)

[2.2.8 SMS 11](#_Toc514845440)

[2.2.9 Reports 12](#_Toc514845441)

[2. 3 Features 13](#_Toc514845442)

[2. 4 Defects 14](#_Toc514845443)

[Chapter 3 15](#_Toc514845444)

[Analysis 15](#_Toc514845445)

[3.1 User Requirements 16](#_Toc514845446)

[3.1.1 Functional Requirements 16](#_Toc514845447)

[3.1.2 Non-Functional Requirements 16](#_Toc514845448)

[3.2 System Requirements 17](#_Toc514845449)

[3.2.1 Functional Requirements 17](#_Toc514845450)

[3.2.2 Non-Functional Requirements: 19](#_Toc514845451)

[3.3 Use Case 20](#_Toc514845452)

[3.4 Use Case Description 21](#_Toc514845453)

[3.4.1 Add hotel use. 21](#_Toc514845454)

[3.4.2 Request add user. 22](#_Toc514845455)

[3.4.3 System configuration. 22](#_Toc514845456)

[3.4.4 Purchases management. 24](#_Toc514845457)

[3.4.5 Accounts management. 25](#_Toc514845458)

[3.4.6 Rooms reservation. 26](#_Toc514845459)

[3.4.7 Generate reports 26](#_Toc514845460)

[3.5 Class Diagram 28](#_Toc514845461)

[Chapter 4 29](#_Toc514845462)

[Design 29](#_Toc514845463)

[4.1 Architecture Design 30](#_Toc514845464)

[4.2 Database Design 31](#_Toc514845465)

[4.3 Sequence Diagram 32](#_Toc514845466)

[4.3.1 Request add use 32](#_Toc514845467)

[4.3.2 Add hotel`s user 33](#_Toc514845468)

[4.3.3 System configuration 34](#_Toc514845469)

[4.3.4 Accounts management 35](#_Toc514845470)

[4.3.5 Purchases management 36](#_Toc514845471)

[4.3.6 Rooms reservation 37](#_Toc514845472)

[4.3.7 Generate report 38](#_Toc514845473)

[4.4 Interface Design 39](#_Toc514845474)

[Chapter 5 43](#_Toc514845475)

[Implementation 43](#_Toc514845476)

[5.1 System code 44](#_Toc514845477)

[5.1.1 Reservations 44](#_Toc514845478)

[5.1.2 Service 46](#_Toc514845479)

[5.1.3 Purchases 48](#_Toc514845480)

[Chapter 6 52](#_Toc514845481)

[Conclusion and future work 52](#_Toc514845482)

[6.1 Conclusion 53](#_Toc514845483)

[6.2 future work 53](#_Toc514845484)

[6.3 References 54](#_Toc514845485)

## List of Tables

[Table ‎3‑1 - Use Case Description/ Configure hotels users 22](#_Toc514967266)

## 

## List of Figures

[Figure ‎0‑1.1 - Methodology Planning Loops 5](#_Toc514966327)

[Figure ‎2‑1 - Nazeel login page 7](#_Toc514966328)

# Chapter 1

# Introduction

## 1.1 Project Title

Hotels Management System.

## 1.2 Introduction

Due to the huge development in the technology and its rapid spread in the last 20 years. Due to this revelation in technology, organizations nowadays tend to use the technology to word this revolution. Most organizations today have huge systems to manage their business processes and these systems work with a huge amount of data so its need to be managed in the right way, and one of these organizations are the hotels.

Hotels Management System is the system that mange all the functions in the hotel like Reservations, rooms cases, services , customers, purchases, Reports, And due to the huge growth in number of hotels nowadays and its need for systems to manage its processes, The need for Hotels Management System had increased. However, Hotel Management systems nowadays are not easy to get or access because its cost a lot and it is not easy to use.

So we decided to build a Hotel`s management system that can be easy to access and use and doesn’t cost a lot.

## 1.3 Problem Statement

* The high cost Hotel`s management system nowadays.
* Hotel`s management system nowadays is not online, so they can`t reach to same Database if they have branches.
* Hotel`s management system nowadays is not easy to use.
* If you want to get any updates in the systems, you have to pay.

## 1.4 Project Objectives

* The system will be cheaper than any other systems, because it`s online and we will put monthly subscription.
* The system will be online, so you can access from anywhere.
* The system will be easy to use.
* Any updates in the system will be free.
* The system will provide the next functions effectively:
* System configuration.
* Reservations.
* Services.
* Purchases.
* Accounts.
* Customers.
* Reports.

## 1.5 Project Scope

**1.5.1 Geographical Scope:**

* This system can be used in any hotel across the world.

**1.5.1 Functional Scope:**

* The system will manage rooms and reservations.
* The system will manage accounts and customers.
* The system will configure services.
* The system will configure purchases.
* The system will be accessed online.
* The system will provide reports.
* Every hotel will have his own database in the server.

## 1.6 Project Methodology

The methodology that will be used in this project is an extreme programming methodology (Agile XP methodology).

The reasons that we choose this methodology is:

* 70% of the project is coding.
* To develop it easily in the future.
* There is not much time to develop the system.

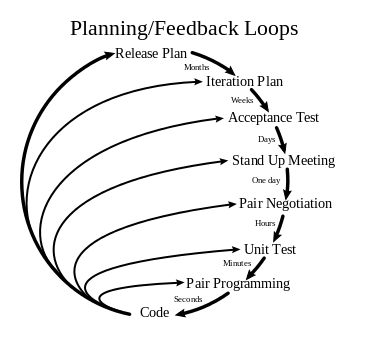
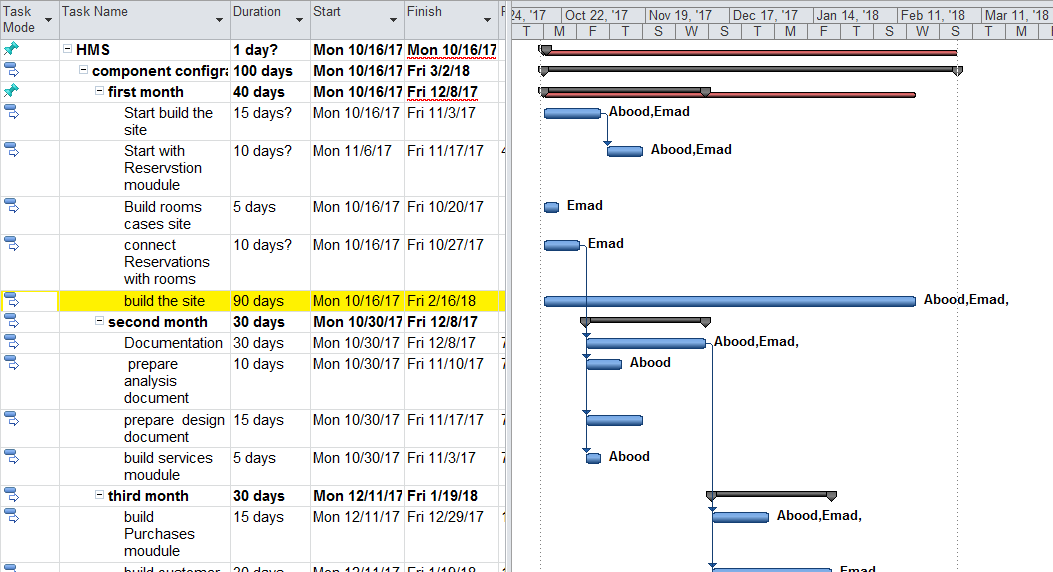
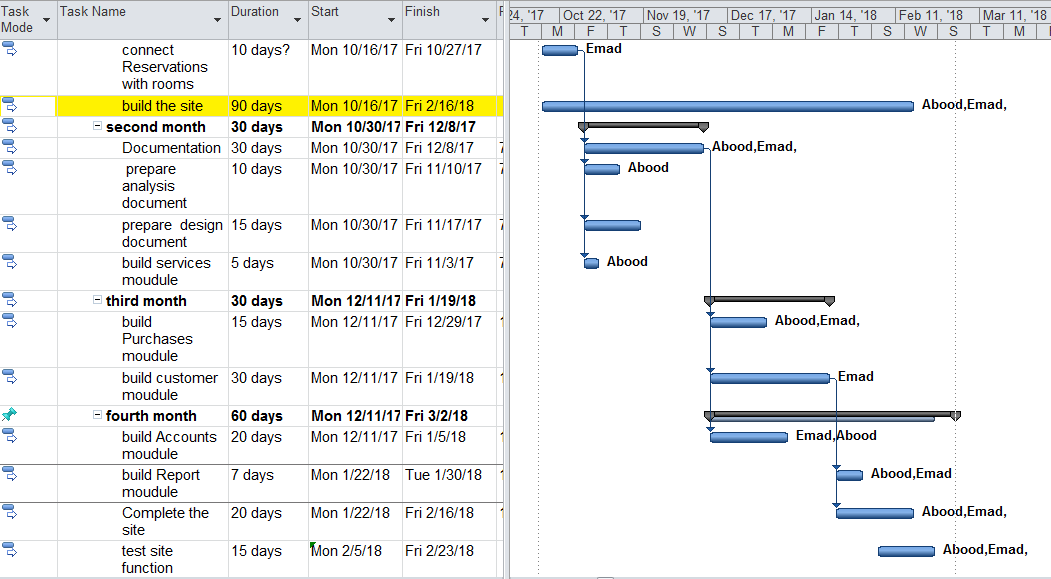


Figure ‎0‑1.1 - Methodology Planning Loops

## 1.7 Schedule Plan





## 1.8 Tools

* NodeJS.
* Visual Studio Code
* MongoDB
* Git
* GitHub
* Google Chrome

# 

# Chapter 2

# Literature review

## Introduction

Since this project is based on building a new system, we will be examining an existing system and try to find its drawbacks. The reason behind this review is to find better and faster solutions for problems to apply it to the new system.

## Nazeel Management System

Nazeel is an online management system for hotels. The system consists of the following:

### Login page



Figure ‎2‑1 - Nazeel login page

### Main page

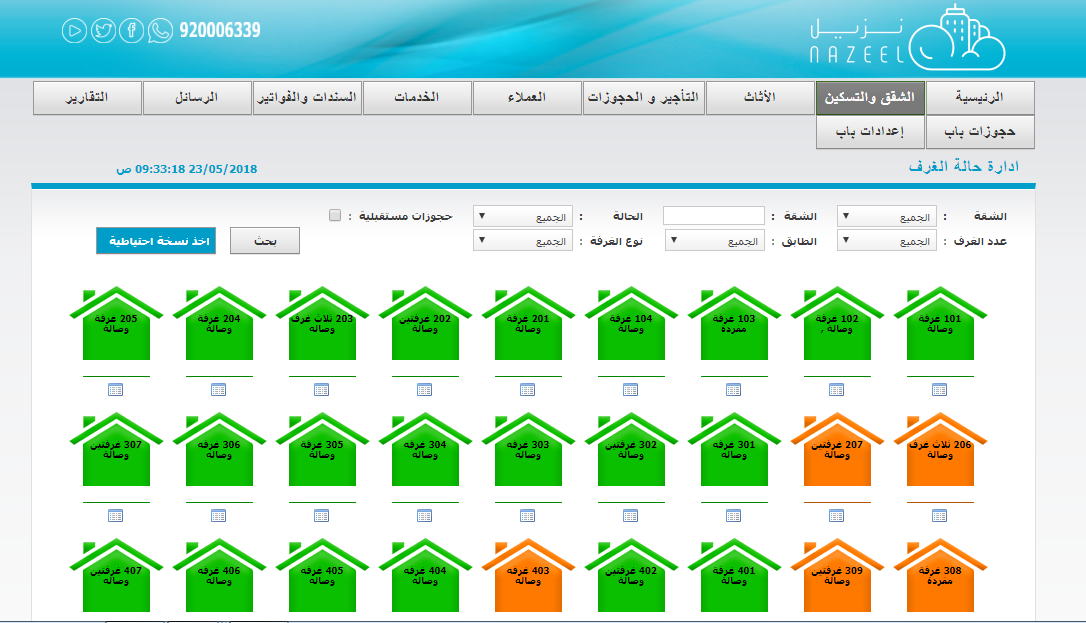


### Reservations

This system allows hotels to:

* Show the admin the available rooms in order to check in.
* Show the admin the rented rooms in order to check out.
* Allows admin to change the states of the rooms.







### Furniture

The system allows furniture tracking for each room inside the hotel. The user can add, modify or remove any peace of a furniture.



### Customers

The system has a customer’s module to manage new or existing customers.



### Services

The system allows admin to order some services and the system will automatically add the charge to the bill of a certain resident.



### Bills

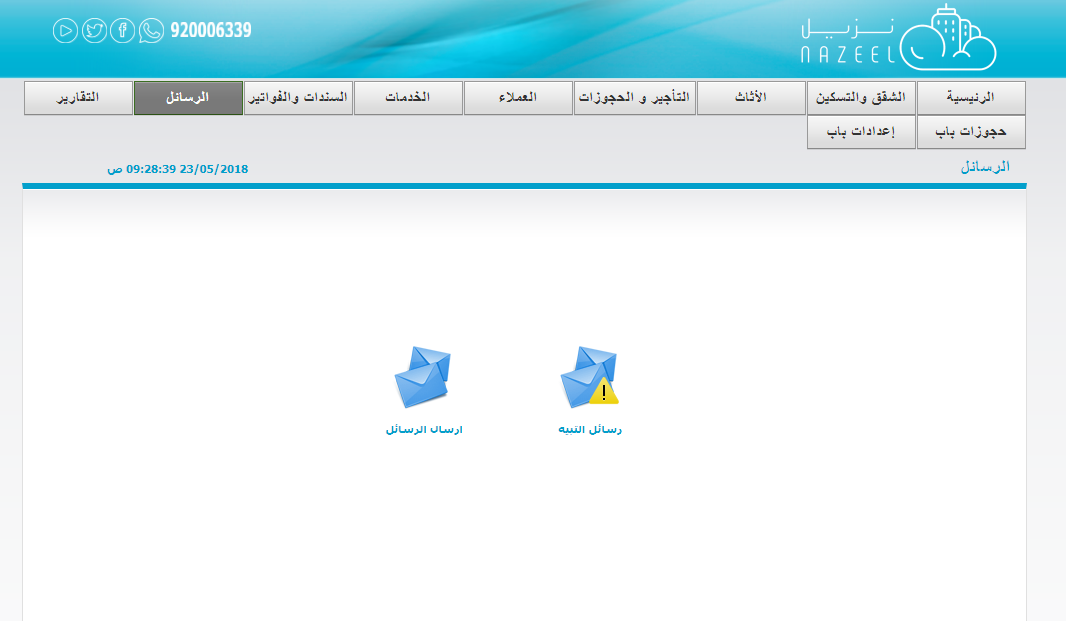
The system has an entire billing module:

* Exchange Bonds
* Banknotes
* Services
* Receipts



### SMS

The system is integrated with a third-party SMS provider to allow hotels to use the service for their customer.



### Reports

The system provides the following reports:

* Total monthly report.
* Bills Report.
* Customer Report.
* Message summary report.
* Reservations report.
* Services report.



## Features

* The system is easy to use.
* The system is easy to get.
* The system is cheap.
* The system is has SMS service.
* The system is has a lot of reports.
* The system is has bills module.

## Defects

* The system is not has a good designed.
* The system is not has purchases module.
* The system is not has a complete services module.

# Chapter 3

# Analysis

* **User requirements** 
  + **Functional Requirements**
  + **Non-Functional Requirements**
* **System Requirements** 
  + **Functional Requirements**
  + **Non-Functional Requirements**
* **Requirements Modelling**
  + **Use Case Diagram**
  + **Use Cases Description**
  + **Class Diagram**

## User Requirements

### Functional Requirements

1. The system must allow the System admin to configure hotels users.
2. The system must allow the Hotel`s admin to configure hotel`s rooms.
3. The system must allow the Hotel`s admin to add Reservation.
4. The system must allow the Hotel`s admin to manage hotel`s Purchases.
5. The system must allow the Hotel`s admin to add hotel`s services.
6. The system must allow the Hotel`s admin to add hotel`s reports and send it to the hotel manager on demand.
7. The system must allow the Hotel`s admin to manage hotel`s accounts.

### Non-Functional Requirements

1. Speed:

The system must be fast in work where each process will take 3 seconds at most.

1. Usability:

The system must be easy to use where is the training time will be three hours at most.

1. High security:

* The system must have high security to protect the System from any external attack.

1. Maintainability:

* The system must be Maintainable to be easy in maintain and to develop the system in the future.

1. Efficiency:

* The system must work without any errors.

## System Requirements

### Functional Requirements

1. The system must allow the System admin to configure hotels users.

* Add user.
* Delete user.
* Update user.
* Show users.

1. The system must allow the admin to configure hotel`s rooms.
2. The system must provide interface to add hotel`s rooms.
3. The system must provide interface to show hotel`s rooms.
4. The system must allow the admin to add Reservation, but before that the admin must add customer's data
5. The system must allow the admin to manage hotel`s Purchases.
6. The system must provide interface to add and show the different type of hotel`s Purchases.

* Furniture.
* Material.
* Others.

1. The system must allow the admin to add hotel`s services.
2. The system must provide interface to add and show the different type of hotel`s services.

* Clean services.
* Maintenance services.
* Food services.
* Other services.

1. The system must allow the admin to add hotel`s reports and send it to the hotel manager on demand.
2. The system must provide interface to add and show the different type of hotel`s reports.

* Reservations report.
* Services report.
* Purchases report.
* Accounts report.
* Users report.
* Customers report.

1. The system must allow the admin to manage hotel`s accounts.

### Non-Functional Requirements:

1. Speed:

Data Transfer between Clint and server will take 3 seconds at most in the middle internet speed by using NodeJS with MongoDB.

1. Usability:

The system must be easy to use where is the training time will be three hours at most.

1. High security:

* The system must have high security to protect the System from any external attack.

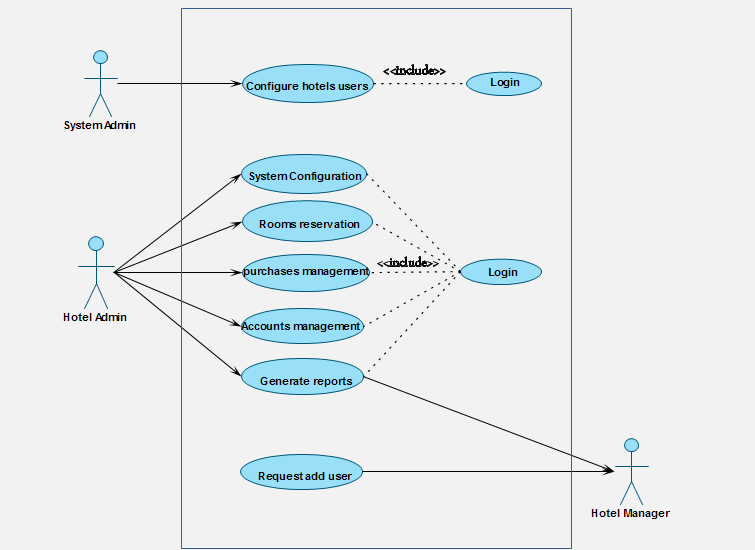
1. Maintainability:

* The system must be Maintainable to be easy in maintain and to develop the system in the future.

1. Efficiency:

* The system must work without any errors.

## Use Case



## Use Case Description

### Add hotel use.

Table ‎3‑1 - Use Case Description/ Configure hotels users

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 1 | Use Case Name: Configure hotels users | |
| Use Case Type: Essential | | | **Primary Actor: System Admin** |
| Stakeholders:  System Admin | | | |
| Brief Description: This use case describes how the system Configure hotels users. | | | |
| Pre-condition: user open the System and request add user. | | | |
| Post-condition : ----- | | | |
| Trigger: user open the System and request add user.  Type: External | | | |
| Relationships:  Include: Login.  Extend: | | | |
| Normal Flow of Events:   1. The user will open the System. 2. The user will request add user [E1]. 3. The System Admin will receive the request. 4. The System Admin will add a new user. | | | |
| Alternate/Exceptional Flows:  [E1] If the user want to see the system before get a user:  1. The System Admin will give him an experimental user to see the system. | | | |

### Request add user.

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 2 | Use Case Name: Request add user | |
| Use Case Type: Essential | | | **Primary Actor: Hotel manager** |
| Stakeholders:  Hotel manager | | | |
| Brief Description: This use case describes how the user Request add user. | | | |
| Pre-condition: user open the System and want to get a user. | | | |
| Post-condition : ----- | | | |
| Trigger: user open the System and request a user.  Type: External | | | |
| Relationships:  Include:  Extend: | | | |
| Normal Flow of Events:   1. The user will open the System. 2. The user want to get a user [E1]. 3. The user will request add user. | | | |
| Alternate/Exceptional Flows:  [E1] If the user want to see the system before add a user:  1. The System Admin will give him an experimental user to see the system. | | | |

### System configuration.

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 3 | Use Case Name: System configuration | |
| Use Case Type: Essential | | | **Primary Actor: Hotel Admin** |
| Stakeholders:  Hotel Admin | | | |
| Brief Description: This use case describes how the admin configure the system. | | | |
| Pre-condition: admin login to the system | | | |
| Trigger: admin open the configuration interface.  Type: internal | | | |
| Relationships:  Include: login. | | | |
| Normal Flow of Events:   1. The admin will login to the system. 2. The admin will open the interface of configuration. 3. The system will show the different kind of configuration. 4. The admin will choose the kind of configuration. 5. if he choose Add hotel`s rooms [A1] , Add hotel`s services [A2] , Add hotel`s purchases [A3] . | | | |
| Alternate Flows:  [A1] Add hotel`s rooms :   * The system will show the interface of Add hotel`s room. * The admin will add hotel`s rooms.   [A2] Add hotel`s services :   * The system will show the interface of Add hotel`s services. * The admin will choose the kind of services. * Clean services. * The admin will add clean services. * Maintenance services. * The admin will add Maintenance services. * Other services. * The admin will add the Other services.   [A3] Add hotel`s purchases :   * The system will show the interface of Add hotel`s purchases . * The admin will Add hotel`s purchases . | | | |

### Purchases management.

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 4 | Use Case Name: purchases management. | |
| Use Case Type: Essential | | | **Primary Actor: Hotel Admin** |
| Stakeholders:  Hotel Admin | | | |
| Brief Description: This use case describes how the admin manage the purchases. | | | |
| Pre-condition: admin login to the system | | | |
| Post-condition : ----- | | | |
| Trigger: admin open the interface of purchases.  Type: internal | | | |
| Relationships:  Include: login.  Extend: | | | |
| Normal Flow of Events:   1. The admin will login to the system. 2. The admin will open the interface of purchases. 3. The admin will choose the kind of purchases. 4. If he choose furniture [A1] , material [A2] , others [A3]. | | | |
| Alternate Flows:  [A1] furniture :   * The admin will add the purchase bonds of furniture.   [A2] material :   * The admin will add the purchase bonds of material.   [A3] others:   * The admin will add the purchase bonds of things that they buy. | | | |

### Accounts management.

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 5 | Use Case Name: Accounts management. | |
| Use Case Type: Essential | | | **Primary Actor: Hotel Admin** |
| Stakeholders:  Hotel Admin | | | |
| Brief Description: This use case describes how the admin manage the accounts. | | | |
| Pre-condition: admin login to the system. | | | |
| Post-condition : ----- | | | |
| Trigger: admin open the interface of accounts.  Type: internal | | | |
| Relationships:  Include: login.  Extend: | | | |
| Normal Flow of Events:   1. The admin will login to the system. 2. The admin will open the interface of accounts. 3. The admin will choose the customer to add bonds. 4. The admin will add bonds to the customer. | | | |

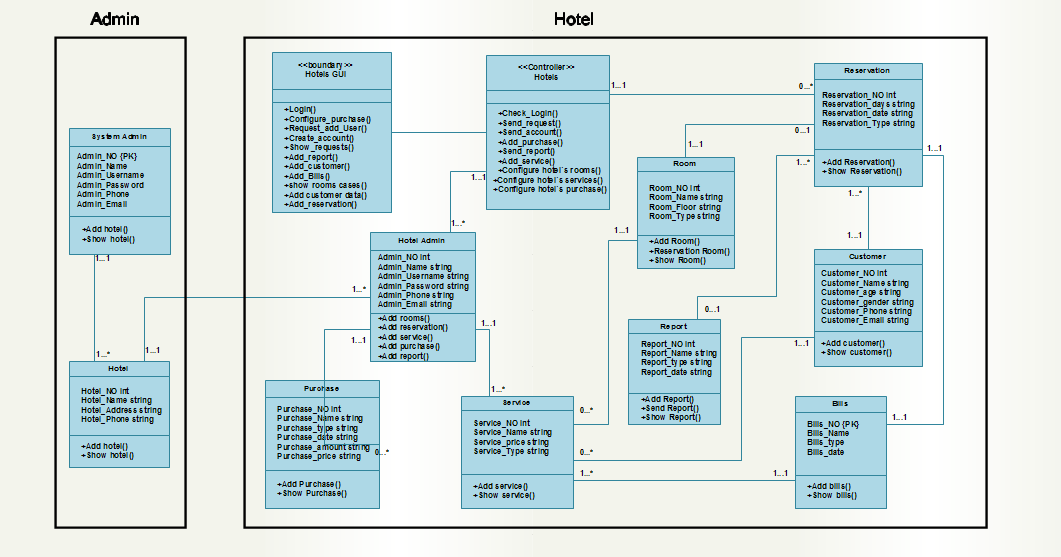
### Rooms reservation.

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 6 | Use Case Name: Rooms reservation. | |
| Use Case Type: Essential | | | **Primary Actor: Hotel Admin** |
| Stakeholders:  Hotel Admin | | | |
| Brief Description: This use case describes how the admin add reservation to rooms. | | | |
| Pre-condition: admin must add customer data. | | | |
| Post-condition : ----- | | | |
| Trigger: admin open the interface of add reservation.  Type: internal. | | | |
| Relationships:  Include: Login.  Extend: | | | |
| Normal Flow of Events:   1. The admin will login to the system 2. The admin will open the interface of show rooms cases. 3. The system will show the rooms cases to know the empty rooms. 4. The admin will search about the specific room. 5. The admin will add customers data. 6. The admin will add reservation to the room. | | | |

### Generate reports

|  |  |  |  |
| --- | --- | --- | --- |
| Importance Level: High | ID: 7 | Use Case Name: Generate reports. | |
| Use Case Type: Details. | | | Primary Actor: Hotel Admin |
| Stakeholders:  Hotel Admin, hotel manager. | | | |
| Brief Description: This use case describes how the admin add and send reports to hotel manager. | | | |
| Pre-condition: hotel manager request reports. | | | |
| Trigger: admin open the interface of add and send reports.  Type: internal. | | | |
| Relationships:  Include: Login. | | | |
| Normal Flow of Events:   1. The admin will login to the system. 2. The admin will open the interface of add reports. 3. The system will show different types of reports. 4. The admin will choose the kind of report. 5. if he choose Reservations report [A1] , Services report [A2] , Purchases report [A3] , Accounts report [A4] , Users report [A5] , Customers report [A6] . | | | |
| Alternate Flows:  [A1] Reservations report:   * The admin will add the Reservations report and send it to the hotel manager.   [A2] Services report:   * The admin will add the Services report and send it to the hotel manager.   [A3] Purchases report:   * The admin will add the Purchases report and send it to the hotel manager.   [A4] Accounts report:   * The admin will add the Accounts report and send it to the hotel manager.   [A5] Users report:   * The admin will add the Users report and send it to the hotel manager.   [A6] Customers report:   * The admin will add the Customers report and send it to the hotel manager. | | | |

## Class Diagram



# Chapter 4

# Design

* **Architecture Design**
* **Database Design**
* **Sequence Diagram**
* **Interface Design**

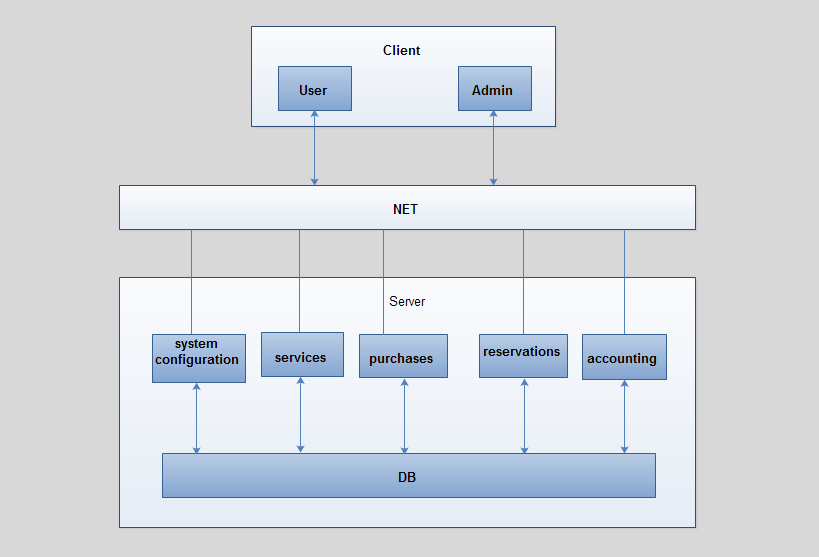
## Architecture Design

The architecture that used is Client-server architecture; we choose this architecture because the system is online.

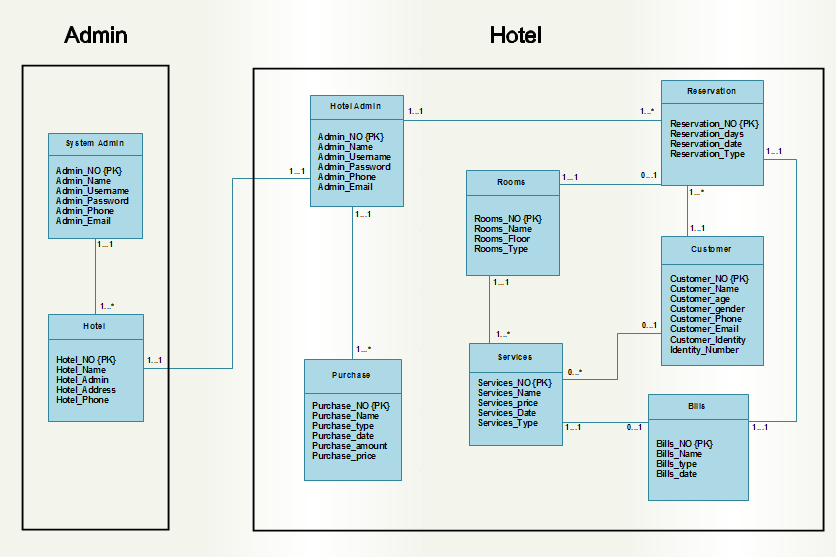
It consists of three sides: client side, server side and internet.

In the client side: that include the admin and the user who is the hotel.

In the server side: that include system configuration, services, purchases, Reservations and Accounts.

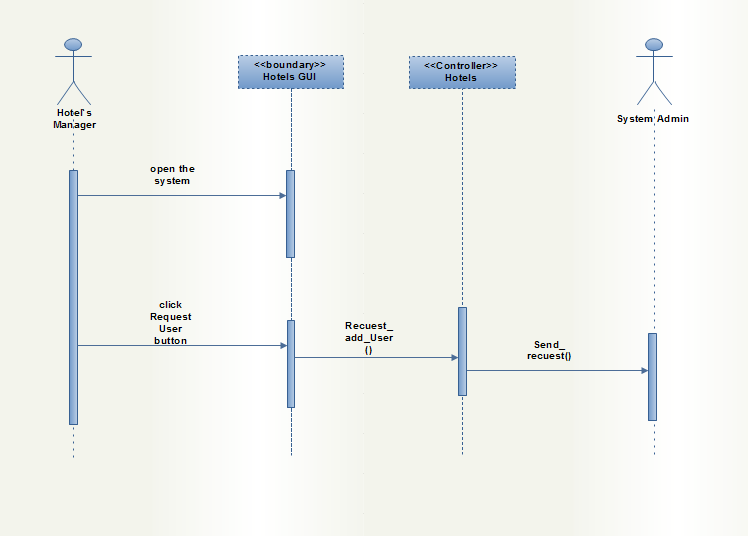


## Database Design

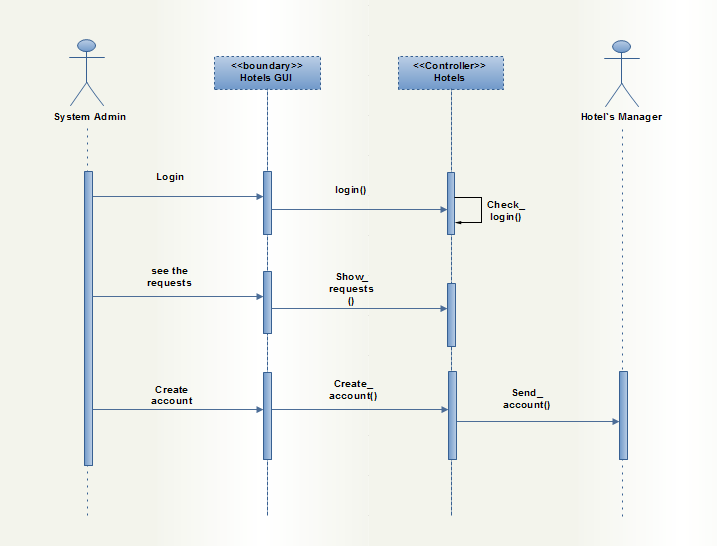


## Sequence Diagram

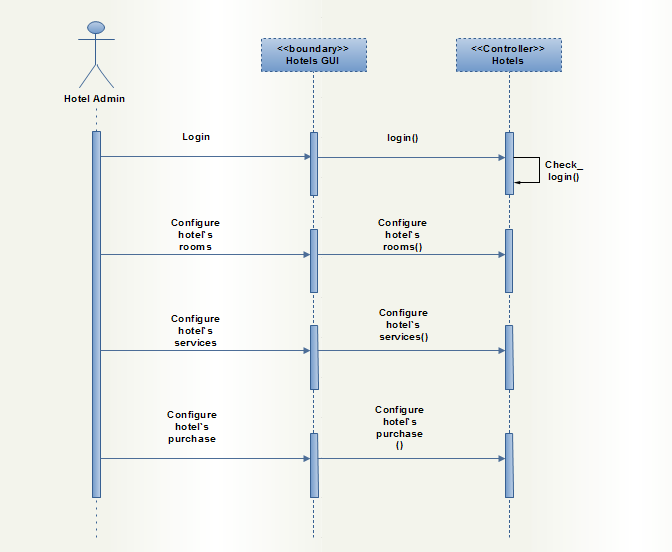
### Request add use



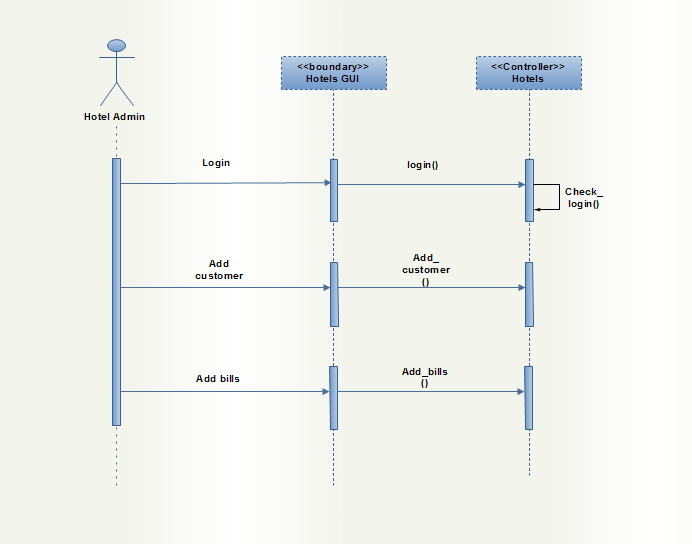
### Add hotel`s user



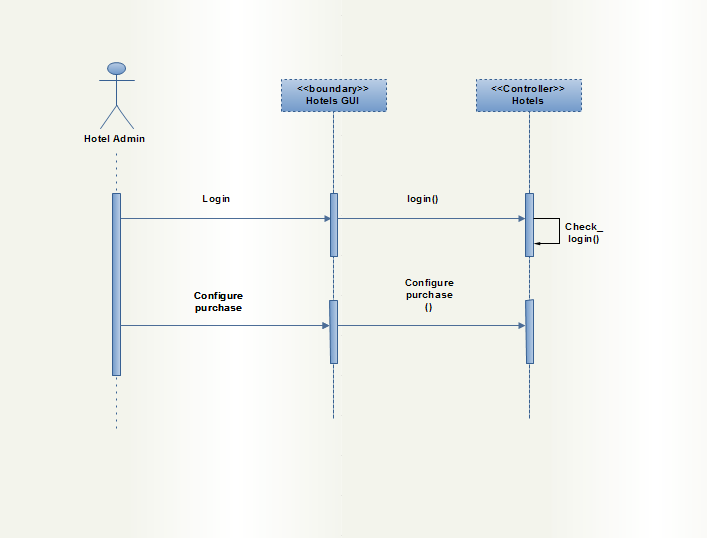
### System configuration



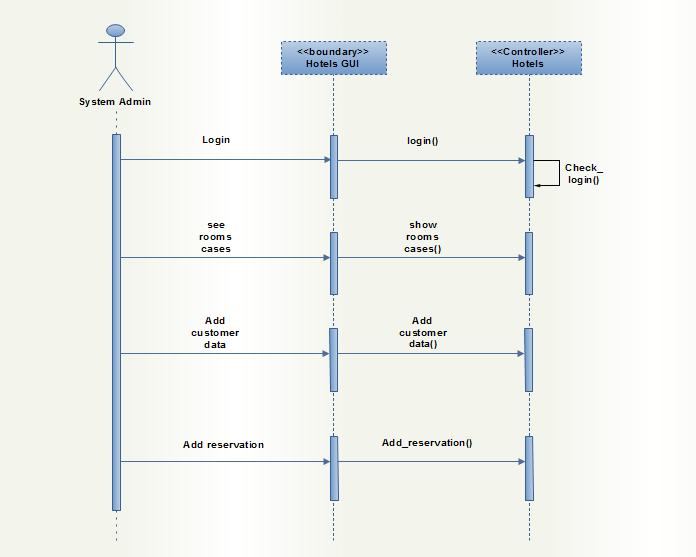
### Accounts management



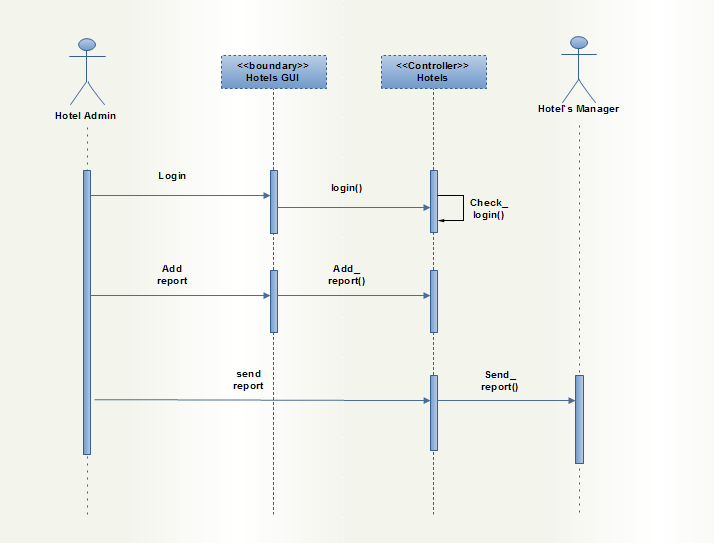
### Purchases management



### Rooms reservation

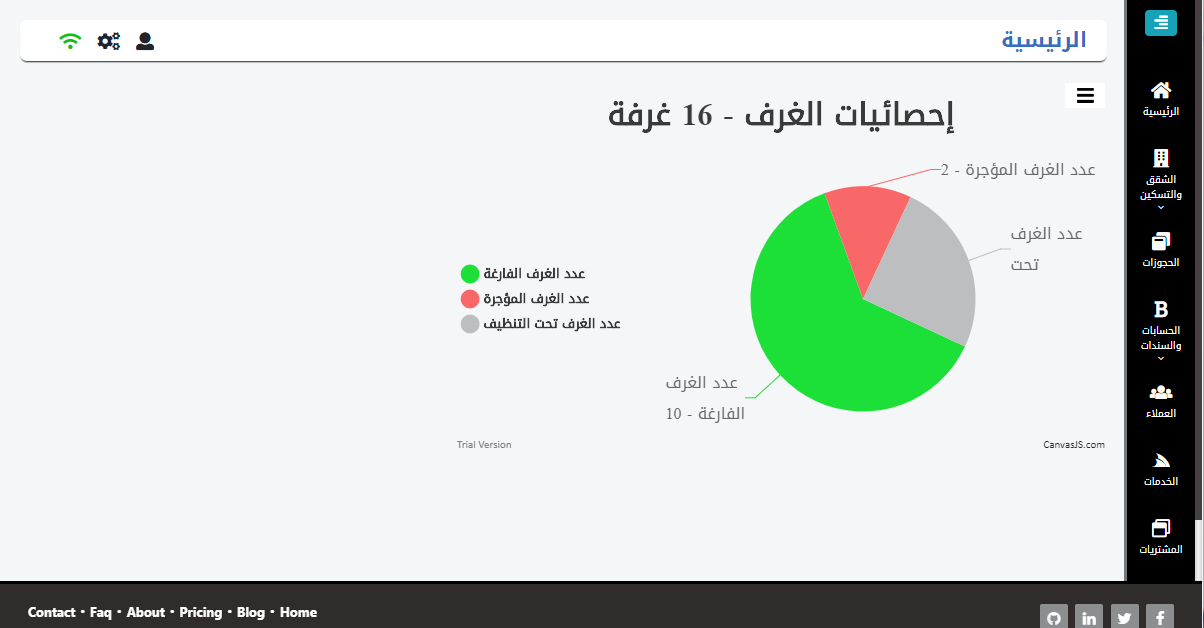


### Generate report



## Interface Design

### The home page



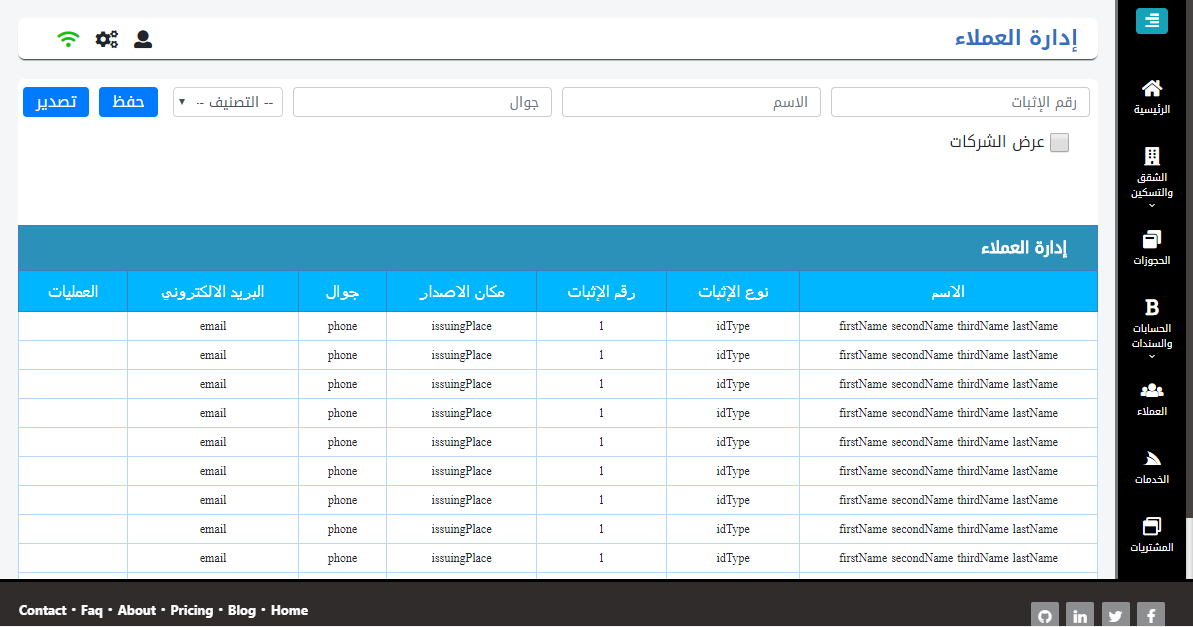
### Manage rooms cases page



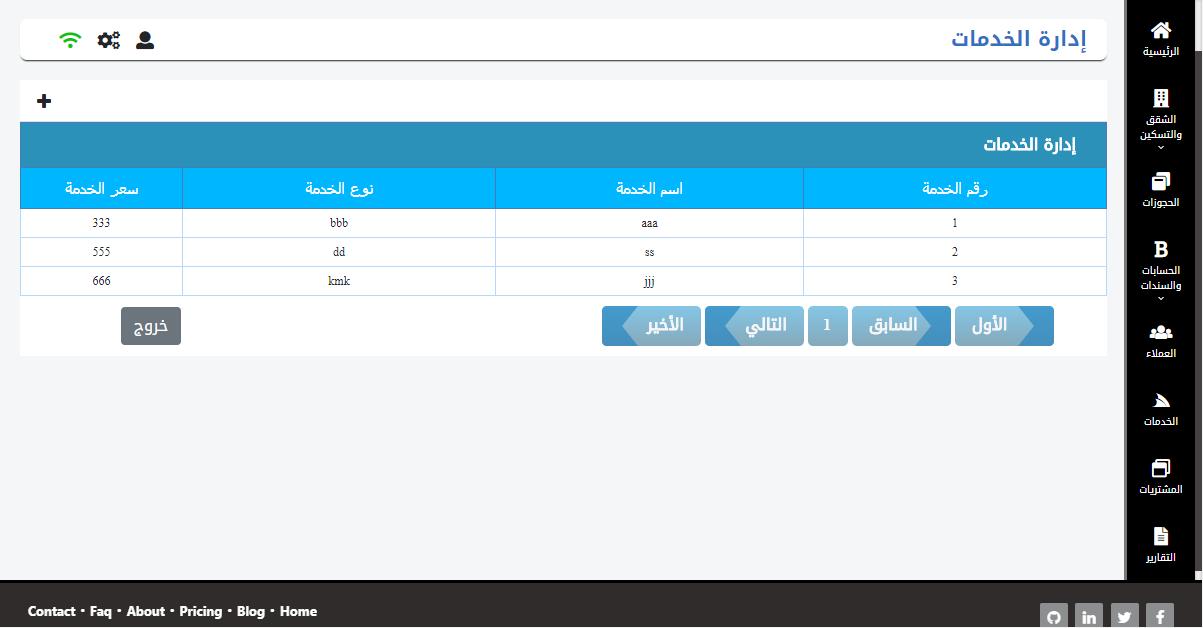
### Reservations page



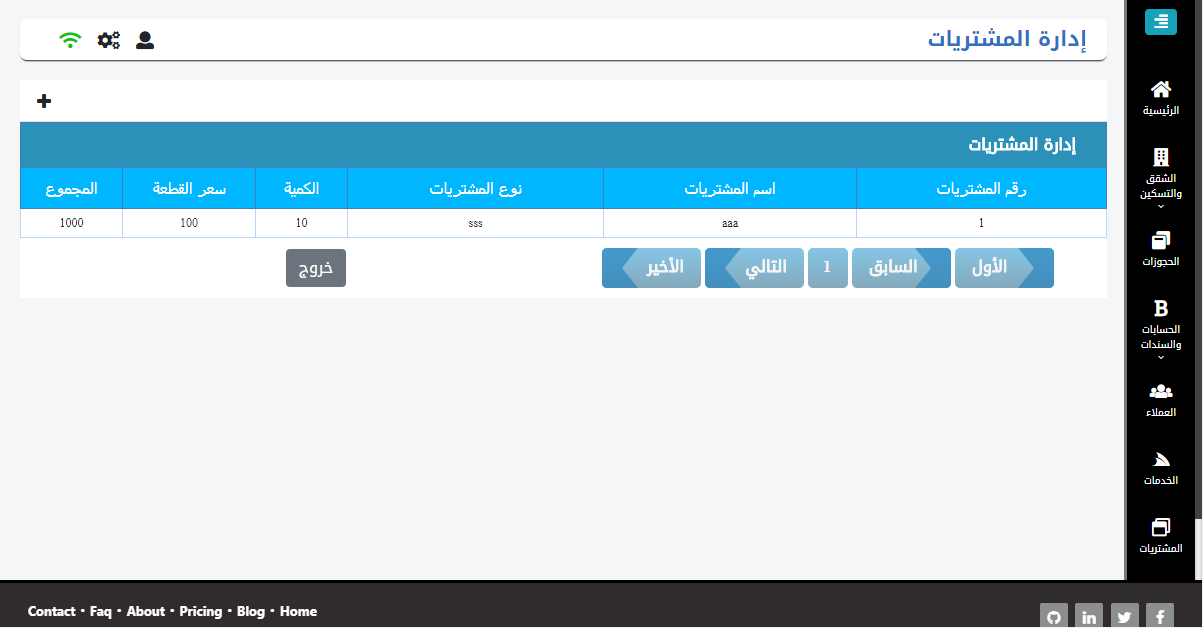
### Customers page



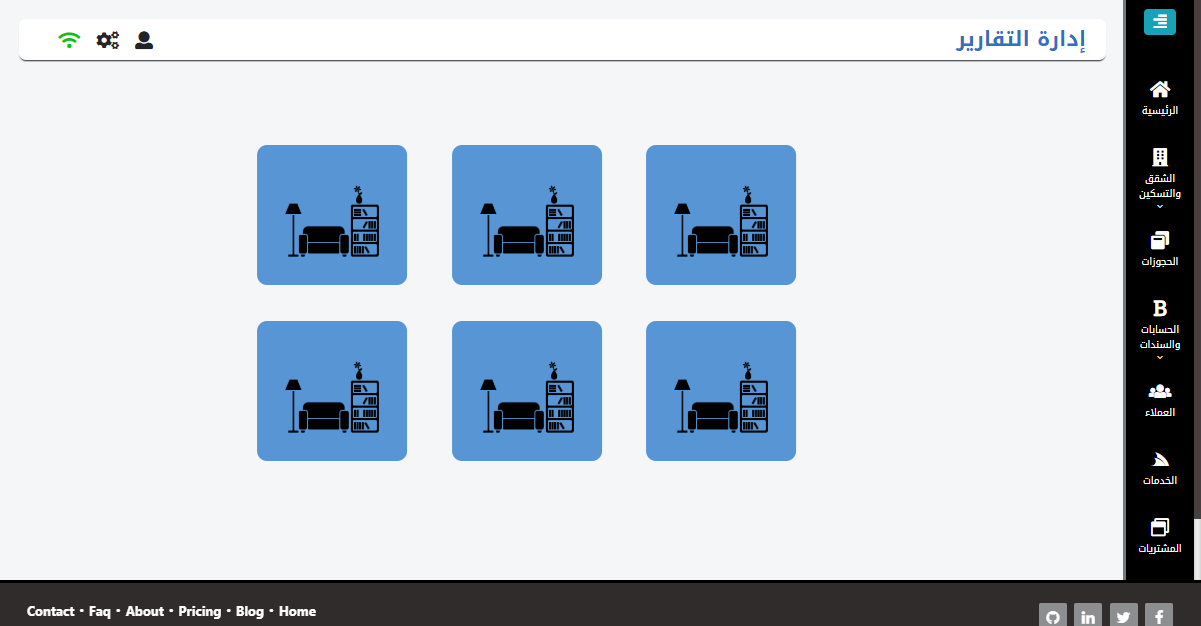
### Services page



### Purchases page



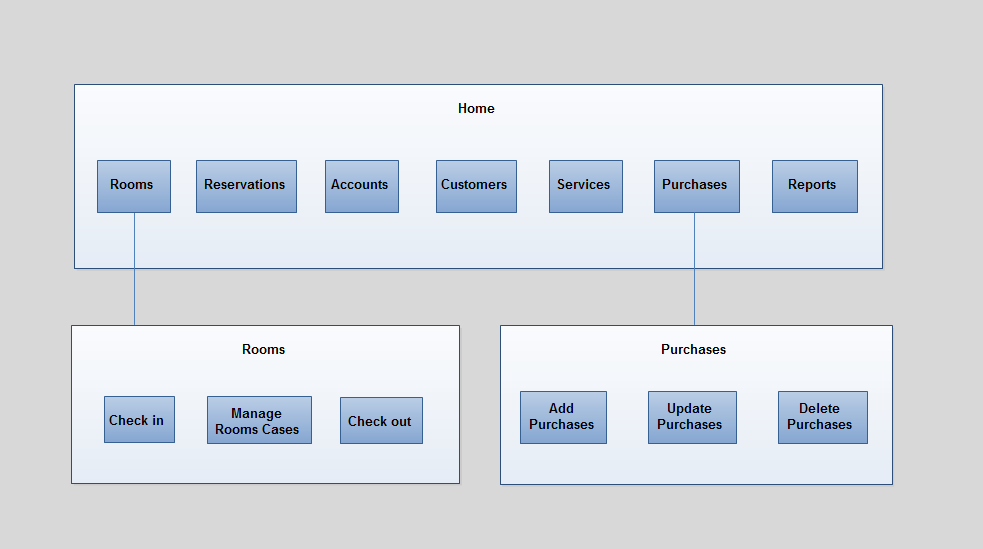
### Reports page

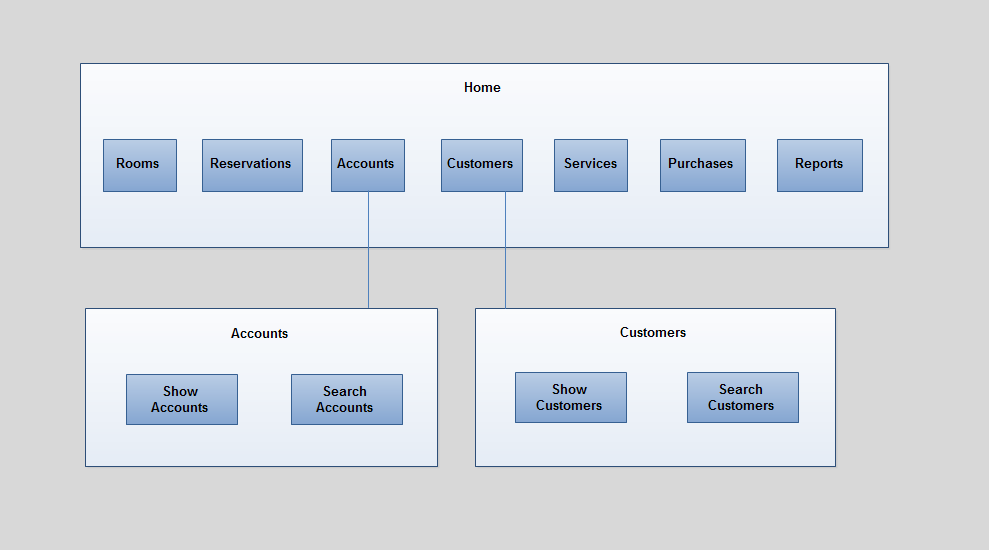


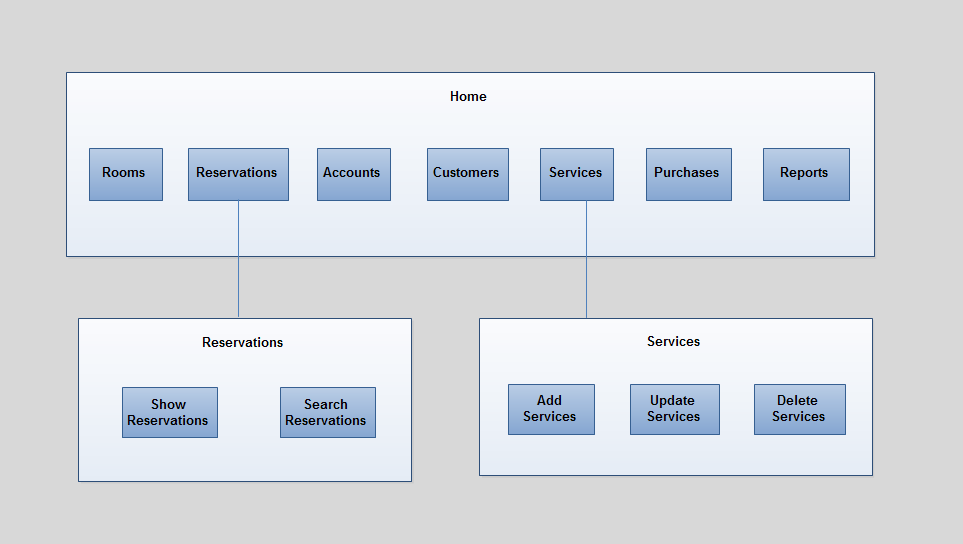
# Chapter 5

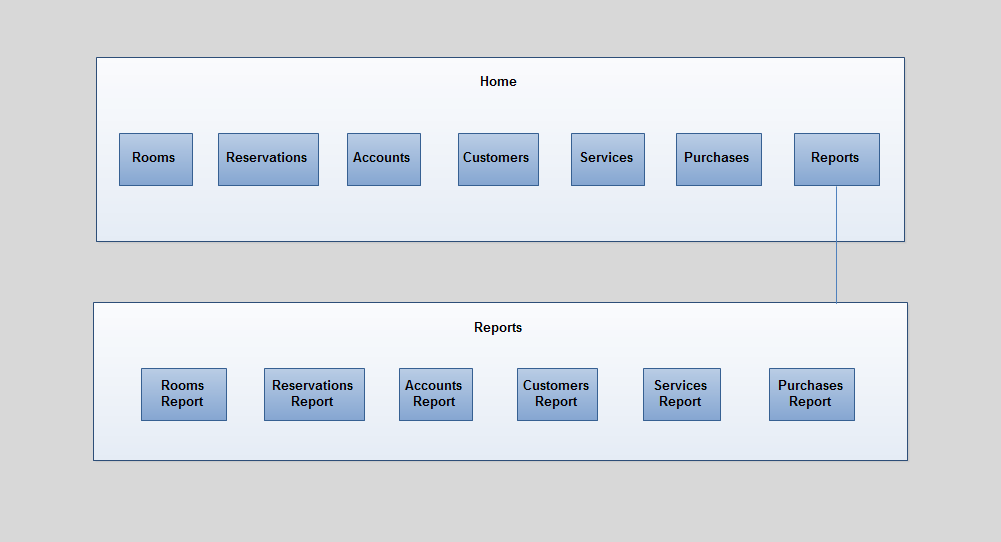
# Implementation

## Site Map









## Component code

### Reservations

var express = require("express");

var router = express.Router();

var MongoClient = require("mongodb").MongoClient;

var globals = require("./globals");

var ObjectId = require("mongodb").ObjectId;

router.get("/",globals.ensureAuthenticated, (req, res, next) => {

MongoClient.connect(globals.url, (err, db) => {

if (err) {

res.render("Pages/reservations", {

title: "الحجوزات",

Page: {

title: "الحجوزات"

},

checkIns: [],

settings: []

});

console.log("Error connecting the database" + err);

} else {

dbo = db.db(globals.dbName);

dbo

.collection("checkIns")

.find({})

.toArray((err, checkIns) => {

if (err) {

console.log("Error connecting the database" + err);

}

dbo

.collection("settings")

.find({})

.toArray((err, settings) => {

if (err) {

res.send(err);

}

res.render("Pages/reservations", {

title: "الحجوزات",

Page: {

title: "الحجوزات"

},

checkIns,

settings

});

});

});

}

});

});

module.exports = router;

### Service

var express = require("express");

var router = express.Router();

var MongoClient = require("mongodb").MongoClient;

var globals = require("./globals");

router.get("/",globals.ensureAuthenticated, (req, res, next) => {

MongoClient.connect(globals.url, (err, db) => {

if (err) {

res.render("Pages/Services", {

title: "الخدمات",

Page: {

title: "الخدمات"

},

services: []

});

console.log("Error connecting the database" + err);

} else {

dbo = db.db(globals.dbName);

dbo

.collection("services")

.find({})

.toArray((err, services) => {

if (err) {

console.log("Error connecting the database" + err);

}

dbo

.collection("settings")

.find({})

.toArray((err, settings) => {

if (err) {

res.send(err);

}

res.render("Pages/Services", {

title: "إدارة الخدمات",

Page: {

title: "إدارة الخدمات"

},

services,

settings

});

});

});

}

});

});

router.post("/addService", (req, res, next) => {

var body = req.body;

var serviceNumber = body.serviceNumber;

var serviceName = body.serviceName;

var serviceType = body.serviceType;

var servicePrice= body.servicePrice;

MongoClient.connect(globals.url, (err, db) => {

if (err) throw err;

dbo = db.db(globals.dbName);

var new\_service = {

serviceNumber: serviceNumber,

serviceName: serviceName,

serviceType: serviceType,

servicePrice: servicePrice,

};

dbo.collection("services").save(new\_service, (err, result) => {

if (err) throw err;

res.redirect("/Services");

// res.end();

db.close();

});

});

});

module.exports = router;

### Purchases

<% include ../partials/header %>

<% include ../partials/sideNav %>

<div class="myContainer">

<div class="content">

<% include ../partials/topNav %>

<section class="wid100" style="background-color:white">

<div class="nav nav-tabs" style="direction: initial">

<a id="btn\_add\_a\_room" class="hint--right hint--rounded nav-item nav-link" role="tab" aria-label="إضافة خدمة" style="cursor:pointer "

data-toggle="modal" data-target="#PurchasesModel">

<i class="fas fa-plus"></i>

</a>

<!-- <label class="fontDroid" style="align-self: flex-end;" >اضافة خدمة</label> -->

</div>

<table id="tblReservations" class="wid100" >

<thead>

<tr class="tblHeader1">

<th colspan="6" class="wid100 fontDroid" style="background-color:#2c91b9;color:white;">إدارة المشتريات</th>

</tr>

<tr class="tblHeader2">

<th>رقم المشتريات</th>

<th>اسم المشتريات</th>

<th>نوع المشتريات</th>

<th>الكمية </th>

<th>سعر القطعة</th>

<th> المجموع</th>

</tr>

</thead>

<tbody>

<% Purchases.forEach(Purchase => { %>

<tr class="tbldatarow">

<td data-title="Name"><%=Purchase.PurchaseNumber %></td>

<td data-title="ID Type"><%=Purchase.PurchaseName %></td>

<td data-title="ID Number"><%=Purchase.PurchaseType %></td>

<td data-title="ID Issue Place"><%=Purchase.Amount %></td> <td data-title="ID Issue Place"><%=Purchase.PurchasePrice %></td>

<td data-title="ID Issue Place"><%=Purchase.PurchasePrice\*Purchase.Amount%></td>

<!-- <td data-title="Actions">

<a id="MainContent\_lstvw\_customers\_btnView\_0" title="View customer" class="ViewIco TblIco" data-href="/Pages/Management/Popups/View/ViewPurchase.aspx?id=3747766&amp;cid=68"

href="javascript:\_\_doPostBack('ctl00$MainContent$lstvw\_customers$ctrl0$btnView','')"></a>

</td> -->

</tr>

<% }) %>

</tbody>

<tfoot>

<tr>

<td colspan="3" class="pd10">

<div class="pagination">

<ul id="MainContent\_dPager2" style="display: inline-flex;">

<li class="navButton" ><a href="#">الأول</a></li>

<li class="navButton"><a href="#">السابق</a></li>

<li class="navButton site"><a href="#">1</a></li>

<li class="navButton back"><a href="#">التالي</a></li>

<li class="navButton back"><a href="#">الأخير</a></li>

</ul>

<div class="PagnTxt"></div>

</td>

<td>

<div class="">

<button id="btnAddRoomClose" type="button" class="btn btn-secondary fontDroid" onclick="location.href='/'">خروج</button>

</div>

</td>

</tr>

</tfoot>

</table>

</section>

</div>

</div>

</div>

</div>

<% include ../partials/footer %>

</body>

<% include ../models/PurchasesModel %>

</html>

<!-- End of Wrapper -->

</div>

### Customer

<% include ../partials/header %>

<% include ../partials/sideNav %>

<!-- <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css"> -->

<!-- <link rel="stylesheet" href="styles/ManageFreeBookings.css">

<link rel="stylesheet"

<div class="myContainer">

<div class="content ">

<% include ../partials/topNav %>

<div class="center3d">

<a data-toggle="modal" data-target="#ReportsModel">

<div class="page-btns reportDiv" id="checkIns" >

<label class="lbl-room\_number">

</label>

<img src="imgs/room.png">

<span class="tooltiptext ">

تقرير الحجوزات

</span>

</div>

</a>

<a data-toggle="modal" data-target="#ReportsModel">

<div class="page-btns reportDiv" id="services" >

<label class="lbl-room\_number">

</label>

<img src="imgs/room.png">

<span class="tooltiptext ">

تقرير الخدمات

</span>

</div>

</a>

</div>

</div>

</div>

<% include ../partials/footer %>

</body>

<% include ../models/ReportsModel %>

<% include ../models/ShowReportModel %>

<script src="https://kendo.cdn.telerik.com/2018.1.221/js/jquery.min.js"></script>

</html>

<!-- End of Wrapper -->

</div>

# Chapter 6

# Conclusion and future work

## Conclusion

Hotels management system online is large system that manage all the functions in the hotel, which has many advantages. This type of Hotels management system will be very helpful to the hotels that want manage all its functions. Where it is can manage room's cases, reserve rooms, manage accounts, add customer data, manage services, manage purchases and generate reports. this system can be accessed from anywhere and it`s very cheap, so any hotel can get it easily.

This project presented Hotels management system that is designed by using NodeJS with MongoDB so that made the system faster and usable. In addition, any update in the system will be free for hotels. The features of this system are: provide easy interfaces for the hotels, easy to use, fast at work, very cheap.

## Future work

In the future, we will try to focus on the development all system. The system should add human resources management module to manage human resources in the hotel, and we will add SMS services.

In the future, we want to complete Accounts module to make it complete.

## References

[1] Nazeel website.

<https://edara.nazeel.net/>