Design Document

**Milton Hotel Website**

**Prepared by**

Waleed Moursy

Parijat Bandyopadhyay

Vinay Govindan

Sreeraj Punnoli

Rishab Gupta

Simon Achkar

Version 1.0

Table of Contents

[1 Introduction 5](#_Toc530321750)

[1.1 Overview 5](#_Toc530321751)

[1.2 Document Purpose 5](#_Toc530321752)

[1.3 Document audience 5](#_Toc530321753)

[1.4 Definitions, Acronyms and abbreviations 6](#_Toc530321754)

[1.5 Reference Document 6](#_Toc530321755)

[2 High-Level Design 7](#_Toc530321756)

[2.1 Goals and Guidelines 7](#_Toc530321757)

[2.2 Architectural Strategies 7](#_Toc530321758)

[2.2.1 User Interface Layer 7](#_Toc530321759)

[2.2.2 Interaction Sequence 8](#_Toc530321760)

[2.2.3 Sign in Screen 10](#_Toc530321761)

[2.2.4 Sign Up Screen 11](#_Toc530321762)

[2.3 High-Level Deployment View 11](#_Toc530321763)

[3 Solution Outline 11](#_Toc530321764)

[3.1 Payment API 11](#_Toc530321765)

[3.1.1 User Interface Layer 11](#_Toc530321766)

[3.1.2 Business Layer 12](#_Toc530321767)

[3.1.3 1.1.3 Data Layer 12](#_Toc530321768)

[3.1.4 API 12](#_Toc530321769)

[3.2 Payment Webpage 13](#_Toc530321770)

[3.2.1 User Interface Layer 13](#_Toc530321771)

[3.2.2 Business Layer 13](#_Toc530321772)

[3.2.3 Data Layer 13](#_Toc530321773)

[3.3 Payment details webpage 13](#_Toc530321774)

[3.3.1 User Interface Layer 14](#_Toc530321775)

[3.3.2 Business Layer 14](#_Toc530321776)

[3.3.3 Data Layer 14](#_Toc530321777)

[3.4 User portal to cancel or edit booking 14](#_Toc530321778)

[3.4.1 User Interface Layer 14](#_Toc530321779)

[3.4.2 Business Layer 15](#_Toc530321780)

[3.4.3 Data Layer 15](#_Toc530321781)

[3.5 Login using Social Media accounts 15](#_Toc530321782)

[3.5.1 User Interface Layer 15](#_Toc530321783)

[3.5.2 Business Layer 15](#_Toc530321784)

[3.5.3 Data Layer 15](#_Toc530321785)

[3.5.4 API 15](#_Toc530321786)

[3.6 Admin Dashboard 16](#_Toc530321787)

[3.6.1 User Interface Layer 16](#_Toc530321788)

[3.6.2 Business Layer 16](#_Toc530321789)

[3.6.3 Data Layer 16](#_Toc530321790)

[3.7 API to show all payments on a date 16](#_Toc530321791)

[3.7.1 User Interface Layer 16](#_Toc530321792)

[3.7.2 Business Layer 16](#_Toc530321793)

[3.7.3 Data Layer 17](#_Toc530321794)

[3.7.4 API 17](#_Toc530321795)

[3.8 Tax rates API 17](#_Toc530321796)

[3.8.1 User Interface Layer 17](#_Toc530321797)

[3.8.2 Business Layer 18](#_Toc530321798)

[3.8.3 Data Layer 18](#_Toc530321799)

[3.8.4 API 18](#_Toc530321800)

[3.9 Email Notifications 18](#_Toc530321801)

[3.9.1 User Interface Layer 18](#_Toc530321802)

[3.9.2 Business Layer 18](#_Toc530321803)

[3.9.3 Data Layer 18](#_Toc530321804)

[3.10 Previous booking review 19](#_Toc530321805)

[3.10.1 User Interface Layer 19](#_Toc530321806)

[3.10.2 Business Layer 19](#_Toc530321807)

[3.10.3 Data Layer 19](#_Toc530321808)

[3.11 A/B Testing 19](#_Toc530321809)

[3.11.1 User Interface Layer 19](#_Toc530321810)

[3.11.2 Business Layer 19](#_Toc530321811)

[3.11.3 Data Layer 19](#_Toc530321812)

[4 Appendix 19](#_Toc530321813)

[4.1 Transaction API 19](#_Toc530321814)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Rishab & Vinay | 11/14/2018 | Initial version | 1.0 |
|  |  |  |  |

# Introduction

## Overview

The project is a hotel website for the customers to view and book the hotel services online. A login is required to make the bookings. If the customer is not an existing user, he will have to sign up using email or one of his social networks. After successful login/signup, the customer will be redirected to his dashboard and can make a search to see the list of available rooms and halls and other events happening in Milton Hotel. The customer can pay online to avail the services via a secure payment gateway. An email will be sent on customer’s email address sending a booking and payment confirmation. An airport pickup can also be scheduled. The hotel also has a restaurant and the menu can be found on the hotel website. The website provides the flexibility to modify/cancel the booking. The customer can view his previous booking and based on those, a series of deals will be shown to the customer. Incorporated in the website is a 24/7 customer support service to assist the customers for queries and inquiries.

The hotel employees have been given different privileges to access the hotel website. From the employee dashboard, they can make entries to the database to update the room availability or change some event details. The employees can additionally access customer’s previous bookings and view their current status to assist them better.

## Document Purpose

This document specifies detailed level design specification of Milton Hotel Website. It provides the detailed level design architectural definition of the solution with the aim of stipulating an optimal implementation of defined and anticipated functional and non-functional requirements while considering the business cost time and quality drivers.

The document must define the information in the relevant section such that the document audience obtains all relevant information about the design specification and consideration. For example, development and testing team should have the understanding to code and prepare the test cases for the design specifications mentioned in this document.

## Document audience

* Development Team - Use as input into development for software component changes.
* Test Team – Ensure understanding of the solution in order to develop test cases and test plans
* Deployment Team – Ensure understanding of how to deploy he solution
* Product Architect – Responsible for ensuring that this document reflects the final solution that is delivered into the live environment.
* Project Manager – To have an overview of the proposed solution
* Product Owner – To have an overview of the proposed solution
* Analyst – To make sure the requirements are met by the solution

## Definitions, Acronyms and abbreviations

|  |  |
| --- | --- |
| **Name** | **Description** |
| API | An application program interface (**API**) is a set of routines, protocols, and tools for building software applications. |
| REST | Representational State Transfer (**REST**) is a software architectural style that defines a set of constraints to be used for creating web services. |
| POST | In computing, **POST** is a request method supported by HTTP used by the World Wide Web. |
| HTTP | Hyper Text Transfer Protocol |

## Reference Document

|  |  |  |
| --- | --- | --- |
| S. No. | Document Name | Version |
| 1 | SRS – Milton Hotel Website | 1.0 |
| 2 | High Level Design – Milton Website | 1.0 |

# High-Level Design

## Goals and Guidelines

The MHW architecture has been designed with the following objectives in mind:

* To allow customers to book rooms, and to edit and cancel their bookings.
* To enable customers to pay online for hotel services and room booking.
* To facilitate user assistance for the hotel's employees by allowing the employees to manage customers' room booking.

The major design and implementation constraints for the system are:

* Simplicity and ease of use
* Flexibility and Scalability

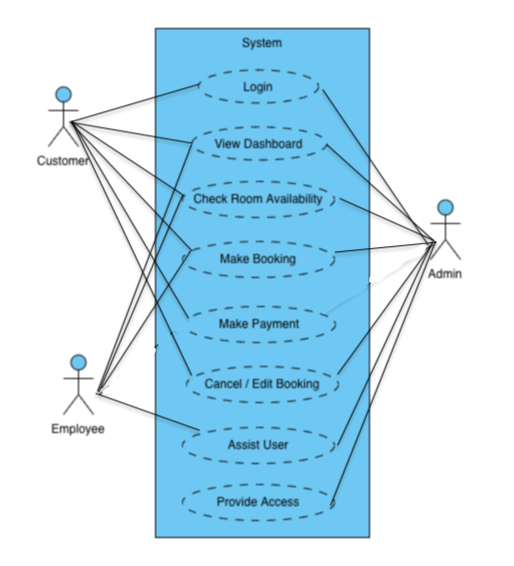
The following sections will explore more deeply how the system works from both the employees and the customers' point of view.

## Architectural Strategies

### User Interface Layer

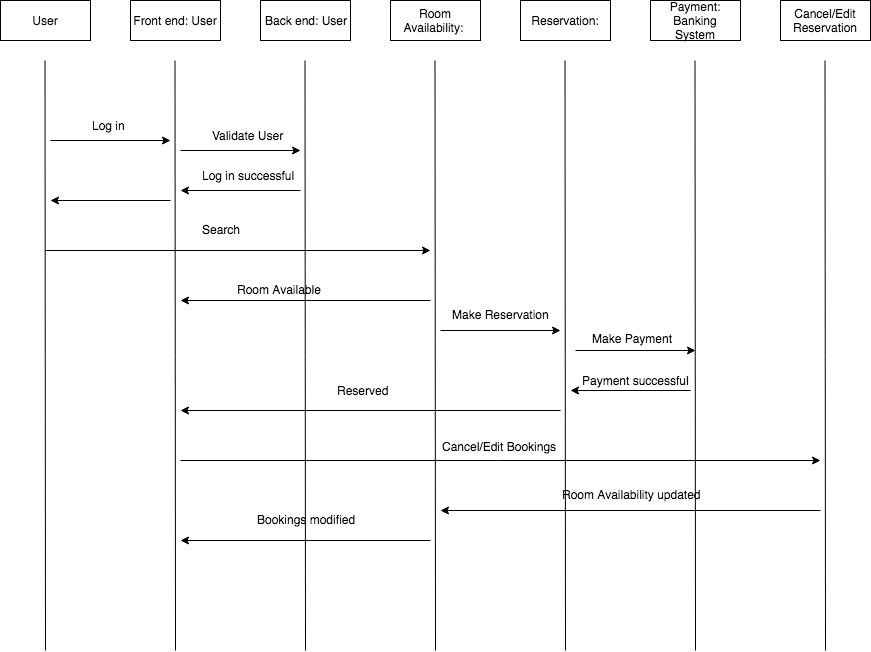
The following is the System’s use case. The three main players are the customers, employees and admin of the system.

The following is the System’s use case. The three main players are the customers, employees and admin of the system.

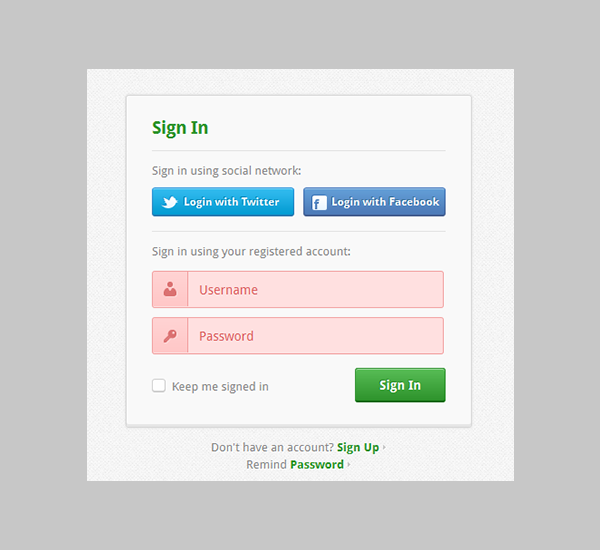


### Interaction Sequence

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



### Sign in Screen



### Sign Up Screen



## High-Level Deployment View

We enter the final deployment view here and we just give an overview

# 

# Solution Outline

## Payment API

It is an API that allows that connects to the bank letting the user make an online payment.

### User Interface Layer

The API has no user interface

### Business Layer

The payment API passes the card/online login details to the bank website while the Transaction API (Bank side) is supposed to get the status of the transaction. A call is made to the transaction API with the transaction number as input, it returns a set of fields as part of the response. The status and the status\_code are the fields that are to be noted down along with the transaction\_id\_bank filed marks the status of the transaction. The filed status has one of two values “Success” or “Failure”. Based on the status code the room is either to be booked or declined and a entry into the Transaction Table and Booking table has to be made.

### 1.1.3 Data Layer

A Transaction table must be created with the following fields Transaction\_id (int primary key), Payload (varchar), request (varchar), response (varchar), endpoint(varchar), Response\_time (double). It is used to track the details of all the transactions that are made through the website. An insert is made into the transaction table based on the request sent to the

### API

**Payment API** call which passes the card/online login details to the bank website.

**End Point :** <http://www.worldbank.com/payment/secure>

**Method :** POST

**JSON Input :**

*{*

*"transaction\_id": Integer,*

*"site": "milton",*

*"client\_name": String,*

*"online\_login": {*

*"login\_id": String,*

*"password": String*

*},*

*"card\_pay": {*

*"card\_number": Integer,*

*"card\_type": String,*

*"cvv" : Integer,*

*"pin" : Integer,*

*"expiry\_date" : String*

*}*

*}*

**Transaction API** call is used to fetch the details of the transaction made.

**End Point :** [http://www.worldbank.com/payment/secure?[id](http://www.worldbank.com/payment/secure?%5bid)]

**Method :** GET

**JSON Response :**

*{*

*"transaction\_id": Integer,*

*"transaction\_id\_bank": Integer,*

*"status": String,*

*"status\_code": Integer,*

*"reason": String,*

*"reason\_code": Integer,*

*}*

## Payment Webpage

The payment webpage is used to enable the customer to make an online payment. It is a form with fields that are required to make the payment.

### User Interface Layer

The payment page appears after the user has selected a room, it has the following fields Payment mode, Bank name, card/account details. If the payment mode is selected as card then the following fields with checks like card number (16 digits), CVV (3 digits), PIN, Expiry date (not lesser than today’s date) are to be displayed for which the input is required. If the payment mode is selected as online banking, then the following fields with checks like login id and password are required. All the values are to be filled and should not be left blank. All the validations are to be done and the error messages are to be displayed before the payment API is called.

After the payment is made the corresponding Success/Failure messages are to be displayed. An option to save the customers payment method is also to be displayed.

### Business Layer

No business layer changes here.

### Data Layer

After the customer enters the details and makes the payments based on the response received an entry is to be made into the Payment history table and the bookings table. This requirement is handled in section 3.12 in this document.

## Payment details webpage

The Payment details webpage is used to display all the payments made by the customer.

### User Interface Layer

The payment details webpage should be accessible from the customers dashboard. When accessed the webpage should display all the payments that the customer has made till date. The customer should be given flexibility to see transactions made between a date range by providing from and to date range boxes. A logout button is to be created in every page.

In order to see the customer dashboard, the customer must login to this account. Hence a Login page with Username and password fields must be created along with a forgot password feature. In case of a new user the user should sign up using the signup page. The signup page should have the following fields, User name (only characters), Email (varchar) which should be a valid and unique email id, Password (hidden), Confirm Password (hidden), Phone Number only numbers of max 10 digits, Address should be displayed. All validations except the unique email id should be made before the values submitted. If the validations are not satisfied, then the corresponding error messages have to be displayed. Once the user successfully signs up then the success message should be displayed, and the user should be directed to the login interface.

### Business Layer

For the payment details webpage, the user payment details must be fetched from the payment history table and bookings table based on the date range selected by the user. If the user doesn’t select the date, then we should display the recent 10 transactions.

For the Signup page a check is supposed to made to see if the email id entered already exists in the customer table and return back an error message if it does.

If the forgot email is clicked, then a check is to be made to see if the customer entered email is valid (i.e.) it exists in the customer table and if it does then it should send a reset password email to the customer.

Sessions are to be created for the customers login a should be dumped upon logout.

### Data Layer

The Customer table is to be created with the following attributes customer id (integer primary key), name (varchar), email (varchar unique), password (varchar encrypted) phone\_number (Integer), Address (varchar).

## User portal to cancel or edit booking

This will help customers to modify their booking and this will bring in flexibility to the customers

### User Interface Layer

This portal should be made accessible from the customer dashboard. This page should display the current bookings that are made by the customer for the future dates. Each booking should be displayed as a list. The following options should be made available corresponding to a booking, a cancel option and a edit option where the user can reschedule or customize the booking. After user either cancels or edits the booking the corresponding status should be displayed

### Business Layer

The future dated bookings and its status should be fetched from the bookings table and should be made into a list and displayed to the customer. If the user edits or cancels a booking the corresponding changes are to be updated in the booking table. Upon canceling of a booking, the refund process must be initiated to the customer and the customer should be provided with credits which he can use for the future booking.

### Data Layer

There is no data layer change here.

## Login using Social Media accounts

The hotel customer would like to use his social media account to sign up and login into his/her hotel account.

### User Interface Layer

In the login page the user must should be provide with an option to signup and login using a social media account. The user should be prompted to select from the list of social media accounts and will be redirected to the corresponding social media site upon selection.

### Business Layer

An API call to the corresponding social media site is to be made which will fetch the required details to enter in to the customer table.

### Data Layer

There are no data layer changes to be made here.

### API

A call to the social media's API should be made in order to fetch the required details to create a new customer entry.

**Endpoint:** Depends on the social media site called

**Method:** GET

**JSON API:**

{

“name” : String,

“date\_of\_birth” : Date,

“password” : String,

“login\_id” : String,

}

## Admin Dashboard

An admin panel / dashboard for the hotel manager or employee with admin privileges. The user of this portal will see a list of all the customers and will have access to all of their public data including their booking history, their payment history, and their order history. This way the admin can help the customer more efficiently if a problem occurs.

### User Interface Layer

A dashboard should be created for the admin which would be different from the customers dashboard. After the admin logs in to webpage, he will be redirected to the admin dashboard. The dashboard contains a list of the users, list of the most recent bookings and services. If the admin clicks on one of the users, they will be redirected to the user’s profile where they can see all the user’s public data including bookings and services provided

The admin when clicked on the room availability link should be allowed to focus on the required room using the search bar and should be capable of booking or editing an existing booking for a customer.

On clicking the authorization page, the admin should be capable of giving access to admin dashboard by creating a login for the employees. This should have same validations and rules as the customer signup page. And only the admin should be capable of making this action.

### Business Layer

Based on the selection made by the admin he should be redirected to the corresponding webpage. If a admin focuses on a customer the details of the user should be fetched from the customer, bookings and payments table to provide the customer with all the information available on the customer.

### Data Layer

The Employee table is the be created with the following attributes SIN (long), Username (varchar), Name (varchar), Password (varchar Encrypted), User\_type (varchar).

## API to show all payments on a date

Admin needs to see all the payments received per date. This will help the admin to analyze the payments received per date

### User Interface Layer

The page to view all the payment should be made available in the dashboard. The admin should be provided a text box to enter the date for which he would make an API call to the bank to view all the payments.

### Business Layer

No Business layer changes required here.

### Data Layer

No Data layer changes required here.

### API

A call should be made to the bank API to show the list of payments received with date as the parameter.

**Endpoint:** [www.royalbank.com/transactions?[account\_number]&[dd-mm-yyyy](http://www.royalbank.com/transactions?%5baccount_number%5d&%5bdd-mm-yyyy)]

**Method:** GET

**JSON Response:**

{

“account\_no” : Integer,

“transcations” : [

“transaction\_1” {

“transaction\_number” : Integer,

“Amount” : Integer,

“Customer\_number” : Integer

},

“transaction\_2” {

“transaction\_number” : Integer,

“Amount” : Integer,

“Customer\_number” : Integer

}

.

.

.

“transaction\_n” {

“transaction\_number” : Integer,

“Amount” : Integer,

“Customer\_number” : Integer

}

]

}

## Tax rates API

The website and Payment API shall be able to handle different tax rates for different customers based on Tax law.

### User Interface Layer

There is no user interface changes here.

### Business Layer

The customers payment is to be modified based on the tax that is fetched using the country of booking.

### Data Layer

There are no data layer changes here.

### API

This will fetch the required tax rate dynamically depending on the country from which the customer is making a booking.

**Endpoint:** [www.taxes.com/calculater?[country](http://www.taxes.com/calculater?%5bcountry)]

**Method:** GET

**JSON Response:**

{

“country” : String,

“country\_code” : Integer,

“federal\_tax” : Float,

“state\_tax” : Float

}

## Email Notifications

Every time a customer makes a payment for the hotel, either to book a room or to buy a specific service. A notification must be sent to the customer’s emailing address informing them of the payment.

### User Interface Layer

There are no user interface changes here.

### Business Layer

The details of the payment and the booking made should be fetched from the booking and the payment table and is to be added in the mail template dynamically. The filled template is to be mailed to the customers email id which is fetched from the customer table. The content of the mail and the time and date of which is was sent is to be backed up into the Email table

### Data Layer

A communications table is to be created with the following attributes email\_id (varchar), email\_type(varchar), date(date), body (varchar) subject (varchar).

## Previous booking review

Hotel customer should be able to review previous booking including dates and payments

### User Interface Layer

On the bookings page make a review option should be provided for the past bookings. Upon clicking it the website must display all the details of the previous booking like Payment made, date of booking, hotel room booked and the duration for which it was booked.

### Business Layer

The booking details should be fetched from the booking and the payment table and should be displayed in a table format.

### Data Layer

No data layer changes required here.

## A/B Testing

The website should support A/B testing in order to support hotel owner in deciding the best view for the website in generating revenue from ads.

### User Interface Layer

The User Interface is to be modified into multiple editions such that the customer can select his preferred view. The ads should be placed at different positions to see which one the customer prefers

### Business Layer

A script to track the most viewed or most preferred view in order to help identify the best view to generate tariff out of adds.

### Data Layer

No data layer changes required here.

## Payment Info to be stored in the database

Create feature to store all the payment information in the database for the admin.

### User Interface Layer

There is no user interface required for this component.

### Business Layer

There is no business layer changes required for this component.

### Data Layer

In A Payment table is to be created here. The Payment table needs the following attributes: **id** (Integer Primary Key), **amount** (long), **currency** (varchar), **payment\_type** (varchar), **details** (varchar), **date** (timestamp). The Bookings table is to be created with the following attributes. A card table is to be created with the following attributes.

# Appendix

## Transaction API

|  |  |
| --- | --- |
| Status Code | Status |
| 22 | Transaction Success |
| 23 | Transaction In progress |
| 11 | Transaction Failed |
| 10 | Transaction not found |
| 12 | Unable to handle request |

|  |  |
| --- | --- |
| Reason Code | Reason |
| 32 | Payment Failure due to invalid Card No. |
| 43 | Payment Failure due to Wrong CVV |
| 55 | Payment Failure due to invalid Expiry Date |