Hotel Tango App

By: Sam Stepter

11/18/2022

Contents

[**Summary Of Changes** 4](#_Toc120007170)

[**Glossary of Terms** 4](#_Toc120007171)

[**System Requirements** 4](#_Toc120007172)

[**Domain Analysis** 4](#_Toc120007173)

[**Customer Statement of Requirements** 4](#_Toc120007174)

[**Glossary of Terms** 5](#_Toc120007175)

[**System Requirements** 6](#_Toc120007176)

[**Functional Requirements** 6](#_Toc120007177)

[**Prioritization Matrix** 6](#_Toc120007178)

[**Non-Functional Requirements** 7](#_Toc120007179)

[**Technical Requirements** 7](#_Toc120007180)

[**C# Classes** 7](#_Toc120007181)

[**C# Web Forms** 7](#_Toc120007182)

[**SQL Server Database** 7](#_Toc120007183)

[**User Interface Requirements** 7](#_Toc120007184)

[**Home Screen** 8](#_Toc120007185)

[**Customer** 8](#_Toc120007186)

[**Room** 10](#_Toc120007187)

[**RoomType** 12](#_Toc120007188)

[**Reservations** 13](#_Toc120007189)

[**Login/Registration** 15](#_Toc120007190)

[**Functional Requirement Specification** 16](#_Toc120007191)

[**Stakeholders** 16](#_Toc120007192)

[**Actors and Goals** 17](#_Toc120007193)

[**Use Cases** 17](#_Toc120007194)

[**Use Case Diagram** 19](#_Toc120007195)

[**Traceability Matrix** 20](#_Toc120007196)

[**Fully Dressed Description** 20](#_Toc120007197)

[**System Sequence Diagrams** 21](#_Toc120007198)

[**Create Customer Sequence Diagram** 21](#_Toc120007199)

[**Create Room Type Sequence Diagram** 22](#_Toc120007200)

[**Create Room Sequence Diagram** 22](#_Toc120007201)

[**Create Reservation Sequence Diagram** 23](#_Toc120007202)

[**User Effort Estimation using Use Case Points** 24](#_Toc120007203)

[**Domain Analysis** 25](#_Toc120007204)

[**Interaction Diagrams** 26](#_Toc120007205)

[**Customer** 26](#_Toc120007206)

[**RoomType** 26](#_Toc120007207)

[**Room** 27](#_Toc120007208)

[**Reservation** 27](#_Toc120007209)

[**Create Reservation** 27](#_Toc120007210)

[**Update Reservation** 28](#_Toc120007211)

[**Get Reservation** 28](#_Toc120007212)

[**Delete Reservation** 29](#_Toc120007213)

[**Class Diagram and Interface Specification** 30](#_Toc120007214)

[**Class Diagram** 30](#_Toc120007215)

[**Traceability Matrix** 30](#_Toc120007216)

[**System Architecture and System Design** 31](#_Toc120007217)

[**Architectural Structure** 31](#_Toc120007218)

[**Mapping Subsystems to Hardware** 31](#_Toc120007219)

[**Persistent Data Storage** 31](#_Toc120007220)

[**Network Protocol** 31](#_Toc120007221)

[**Algorithms and Data Structures** 31](#_Toc120007222)

[**Algorithms** 31](#_Toc120007223)

[**Data Structure** 31](#_Toc120007224)

[**User Interface Design and Implementation** 31](#_Toc120007225)

[**Design of Tests** 31](#_Toc120007226)

[**History of Work, Current Status, and Future Work** 32](#_Toc120007227)

[**References** 32](#_Toc120007228)

# **Summary Of Changes**

## **Glossary of Terms**

* Add RoomType

## **System Requirements**

* Remove Login Class, I opted to use the logins libraries from .NET Core, it was much easier and more secure to leverage than a custom created class. Authentication is now handled by .NET Core libraries.
* Removal of the WIFI class, this will be integrated into the Reservation class and randomly generated by the database.
* **Functional requirement 8**  - The Hotel Tango app must be able to generate a unique WIFI code per customer.
  + Also added to prioritization matrix.
* **Functional requirement 9**  - The Hotel Tango app must allow the creation of unique credentials for all users.
  + Also added to prioritization matrix.
  + Added to traceability matrix.
  + Added to fully dressed description
* **Functional requirement 10**  - The Hotel Tango app must have automated email for new user creation.
  + Also added to prioritization matrix.
  + Added to traceability matrix.
  + Added to fully dressed description.

## **Domain Analysis**

* Remove WIFI Class from diagram, this has been moved into the reservation class.
* Add WIFI\_Passcode to Reservation class.
* Remove Login Class, this will be handled by .NET libraries.

# **Customer Statement of Requirements**

Currently, Hotel Tango owns and manages a property of 50 rooms along with a guest Wi-Fi

management system along with inventory and booking. As it stands, employees of Hotel Tango must

manually maintain hotel reservations, inventory, and the generation of WIFI passwords for new

reservations. This is a tedious amount of work and Hotel Tango wants a more automated solution for

managing the tasks.

Hotel Tango wants a software suite to manage the above issues due to the amount of time it takes to

create a reservation. As it stands, employees spend an average of twenty minutes to create a single reservation

and issue causing a loss in revenue due to slow customer service. Hotel Tango has also experienced loss of

revenue due to overbooking reservations, this scenario tends to occur because of complex record keeping due to a

lack of a centralized reservation system.

Not all Hotel Tango employees are trained in managing the WIFI system, this has created a bottleneck for

certain employees who are trained, as they are the only employees comfortable with the system. Hotel Tango

hopes that having the software suite govern WIFI access will lead to a more streamlined process for both

employees and customers.

Though Hotel Tango is not requesting this feature in the current version of their software, they would like

to query data metrics and view their overall revenue for the month and year. In addition to this reporting, they

would like to view the most booked type of rooms. Having this data would help Hotel Tango decide if their current hotel setup is conducive to their business.

Hotel Tango hopes that the new software suite will provide a single source of truth for their business

transactions. Having a single source of truth will lead to increased revenue and hopefully alleviate employee

workload. Besides providing a more streamlined experience for their employees, Hotel Tango hopes that the

software suite will lead to higher customer satisfaction, as reservations should occur more quickly and accurately.

# **Glossary of Terms**

Customer – This will be composed of basic customer information; this will help Hotel Tango identify who

their customers are, along with limited personal information.

Login – This will be the login data that will be stored and used for authentication by Hotel Tango

employees.

Reservation – This will be composed of a customer data, their room information, cost, date of stay and

Wi-Fi information.

RoomType – This will be composed of hotel room details. Each hotel room will be of a certain type of room containing unique attributes.

Room – As Hotel Tango has several types of rooms, this will be the customers chosen domicile for the

duration of their stay.

WI-FI – This will be a part of the customers reservation, each instance of a WI-FI password will be linked

to a given reservation.

# **System Requirements**

## **Functional Requirements**

|  |  |
| --- | --- |
| **Req #** | **Description** |
| Req 1 | * The Hotel Tango app must be able to add, modify, remove, sort, filter and view all reservations. |
| Req 2 | * The system must be auditable and can track all system changes. |
| Req 3 | * The Hotel Tango app must allow Hotel Tango employees to add, modify, remove, sort, filter and view reservation information |
| Req 4 | * The Hotel Tango app must be able to add, modify, remove, sort, filter and view all customer information. |
| Req 5 | * They system must be able to be restored to a specific point-in-time. |
| Req 6 | * The database must be able to be viewed in the event that Hotel Tango wants to view raw data. |
| Req 7 | * The Hotel Tango app must allow Hotel Tango employees to add, modify, remove, sort, filter and view hotel rooms and room types. |
| Req 8 | * The Hotel Tango app must be able to generate a unique WIFI code per customer. |
| Req 9 | * The Hotel Tango app must allow the creation of unique credential for all users. |
| Req 10 | * The Hotel Tango app must have automated email for new user creation. |

## **Prioritization Matrix**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Req # | Importance to customer  Rate 1-5  High = 5  Low = 1 |  | Likelihood of success  Rate 1-5  High = 5  Low = 1 |  | Cost Reduction  Rate 1-5  High = 5  Low = 1 |  | Leverage (Positive impact on other processes)  Rate 1-5  High = 5  Low = 1 |  | Total Project Priority  Rate 1-5  High = 5  Low = 1 |
| Req 1 | 5 | + | 5 | + | 5 | + | 5 | = | 20 |
| Req 2 | 3 | + | 5 | + | 1 | + | 2 | = | 11 |
| Req 3 | 4 | + | 4 | + | 5 | + | 5 | = | 18 |
| Req 4 | 5 | + | 5 | + | 5 | + | 5 | = | 20 |
| Req 5 | 5 | + | 5 | + | 1 | + | 1 | = | 12 |
| Req 6 | 2 | + | 5 | + | 1 | + | 2 | = | 10 |
| Req 7 | 5 | + | 5 | + | 5 | + | 3 | = | 18 |
| Req 8 | 5 | + | 5 | + | 2 | + | 5 | = | 17 |
| Req 9 | 5 | + | 5 | + | 0 | + | 5 | = | 15 |
| Req 10 | 5 | + | 5 | + | 0 | + | 5 | = | 15 |

## **Non-Functional Requirements**

* The Hotel Tango app must be easy to use.
* The Hotel Tango app must be easily deployed.
* The system must be fast.
* The home page must contain a picture of Hotel Tango.
* The system must minimize licensing costs.
* The Hotel Tango app must allow for simultaneous action from multiple users at a given time.
* The Hotel Tango app must include a privacy statement.

## **Technical Requirements**

### **C# Classes**

* Customer
  + This class will hold customer information and will be sent/returned from the database and presented to a webform.
* Reservation
  + This class will hold reservation information and WIFI code, this information will be sent/returned from the database and presented to a webform.
* Room
  + This class will hold room information and will be sent/returned from the database and presented to a webform.
* RoomType
  + This class will hold all of the attributes of a given room type. This information will be sent/returned from the database and presented to a web form.

### **C# Web Forms**

* Customer Web Form
* Login Web Form
* Reservation Web Form
* Room Web Form
* Room Type Web Form

### **SQL Server Database**

* This database will be used to store structured data and interact with the Hotel Tango app.

## **User Interface Requirements**

* Home Screen must contain a photo of Hotel tango, login info and a nav bar.
* All web forms must have a search box and sort button.
* User must authenticate prior to being able to commit actions in all web forms.

### 

### **Home Screen**

A picture containing graphical user interface

Description automatically generated

### **Customer**

#### **Index**

Graphical user interface, text

Description automatically generated

#### **Create**

Graphical user interface, application

Description automatically generated

#### **Edit**

Graphical user interface, application, table

Description automatically generated

#### **Delete**

Graphical user interface, text, application, email

Description automatically generated

### **Room**

#### **Index**

**Graphical user interface

Description automatically generated**

#### **Edit**

Graphical user interface, text, application, email

Description automatically generated

#### **Delete**

Graphical user interface, text, application, email

Description automatically generated

#### **Create**

Graphical user interface, text, application, email

Description automatically generated

### **RoomType**

#### **Index**

A picture containing table

Description automatically generated

#### **Create**

Graphical user interface, text, application, email

Description automatically generated

#### **Edit**

Graphical user interface, text, application

Description automatically generated

#### **Delete**

**Graphical user interface, text, application, email

Description automatically generated**

### **Reservations**

#### **Index**

Graphical user interface, application

Description automatically generated

#### **Create**

Graphical user interface, text, application

Description automatically generated

#### **Edit**

Graphical user interface, text, application, email

Description automatically generated

#### **Delete**

Graphical user interface, text, application, email

Description automatically generated

### **Login/Registration**

#### **Register**

Graphical user interface, text, application, email

Description automatically generated

#### **Login**

Graphical user interface, text, application, email

Description automatically generated

## **Functional Requirement Specification**

### **Stakeholders**

Below is a list of stakeholders that have vested interest in this application.

* Hotel Owner
* Hotel Management
* Hotel Reservation Concierge employees
* Hotel Customer Service employees

## **Actors and Goals**

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Hotel Owner | * Query raw data to understand data trends. * View reservations for customers. * View rooms available and occupied for the hotel. |
| Hotel Management | * Manage the modification of hotel room rates. * Manage the modification of hotel room types. * Manage the creation and modification of customer data. * Manage the creation and modification of Wi-Fi passwords. * Manage the creation and modification of reservations. * View reservations for customers. * View rooms available and occupied for the hotel. |
| Hotel Concierge | * View reservation information for customers. * View room information for given customers. * Assist customers with WIFI access |
| Hotel Customer Service Employees | * Manage the creation and modification of customer data. * Manage the creation and modification of Wi-Fi passwords. * Manage the creation and modification of reservations. * View reservations for customers. * View rooms available and occupied for the hotel. |

## **Use Cases**

|  |  |
| --- | --- |
| **System Requirement** | **Description** |
| REQ-1: Login | All users will have to login via a valid login and password. |
| REQ-2: Forgot password feature | This will allow users to reset their password in the event they forgot it. |
| REQ-2: View all reservations | This will be a view that allows users to view all current and past reservations. |
| REQ-3: Sort and filter all reservations | As data grows, it will be important for users to have the capability to sort and filter reservations. |
| REQ-4: Create, sort, and modify reservations | There will be separate pages for the following functions:   * Create Reservations * Edit Reservations * Delete Reservations |
| REQ-5: Create, sort, and modify customer data | There will be separate pages for the following functions:   * Create Customers * Edit Customers * Delete Customers |
| REQ-6: Create, sort, and modify WIFI Codes | The creation of a WI-FI code will be randomly generated. There will be a view for modifying a WI-FI code. |
| REQ-7: Create, sort, and modify rooms | There will be separate pages for the following functions:   * Create Rooms * Edit Rooms * Delete Rooms |
| REQ-7: Create, sort, and modify room types | There will be separate pages for the following functions:   * Create Room types * Edit Room types * Delete Room types |

## **Use Case Diagram**

Diagram

Description automatically generated

## **Traceability Matrix**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Req ID |  | Requirement | Test Case ID | Test Case Description | Status |
| 1 |  | Login | TC01 | Create an account | Done |
|  | TC02 | Reset password | Done |
|  | TC03 | Login with Valid Creds | Done |
| 2 |  | **CRUD:** Reservation Information | TC04 | Update Reservation | Done |
|  | TC05 | Delete Reservation | Done |
|  | TC06 | Create Reservation | Done |
|  | TC07 | View Reservation | Done |
|  | TC08 | Sort Reservations | Done |
| 3 |  | **CRUD:** Customer Information | TC09 | Update Customer | Done |
|  | TC10 | Delete Customer | Done |
|  | TC11 | Create Customer | Done |
|  | TC12 | View Customer | Done |
|  | TC13 | Sort Customer | Done |
| 4 |  | **CRUD:** Room Information | TC14 | Update Room Info | Done |
|  | TC15 | Delete Room Info | Done |
|  | TC16 | Create Room Info | Done |
|  | TC17 | View Room Info | Done |
|  | TC18 | Sort Room Info | Done |
| 5 |  | **CRUD:** RoomType Information | TC16 | Update RoomType Info | Done |
|  | TC17 | Delete RoomType Info | Done |
|  | TC18 | Create RoomType Info | Done |
|  | TC19 | View RoomType Info | Done |
|  | TC20 | Sort RoomType Info | Done |
| 6 |  | **CRUD:** WI-FI Information | TC21 | Update WI-FI Info | Done |
|  | TC22 | Delete WI-FI Info | Done |
|  | TC23 | Create WI-FI Info | Done |
|  | TC24 | View WI-FI Info | Done |
|  | TC25 | Sort WI-FI Info | Done |
| 7 |  | Automate Reservation Email | TC26 | Create Reservation test email | Done |
| 8 |  | Create unique credentials | TC27 | Create a new user account | Done |

## **Fully Dressed Description**

**REQ-1 Login:** This requirement allows you to create a user account, reset the password and check that credentials are valid before being authorized to perform system actions. This is important as we want to ensure that the system properly governs who can access all pertinent hotel data.

**REQ-2 CRUD: Reservation Information:** This requirement will allow users to create reservations for customers after their data has been entered into the system. If a reservation is present, it is important to have the ability to view, modify, delete and create any given reservation

**REQ-3: CRUD: Customer Information:** Before creating a reservation, a customer’s data must exist. This requirement will allow users to create a customer profile. After their data has been entered into the system, the system users may proceed in creating a reservation for said user. It is important to have the ability to view, modify, delete and create any given customer profile.

**REQ-4 CRUD: Room Information:** Hotel Tango has several rooms and each of those rooms is a type of room. If Hotel Tango decides to change the layout of their rooms, add rooms, or remove them, it is important for Hotel Tango to be able to modify the rooms. This requirement will allow Hotel Tango to view, add, remove, and modify room information.

**REQ-5 CRUD: RoomType Information:** Each room has a type, whether it be a single or double-bedded hotel room. This requirement will allow Hotel Tango to add, modify, view, and delete all room types. This requirement is necessary so that Hotel Tango books the correct room for a given customer.

**REQ-6 CRUD: WI-FI Information:** Each reservation will contain a WI-FI code. When a user creates a reservation, the system will automatically generate a random string that will serve as the WI-FI code for a given user. This requirement will require that the system automatically generates said code but allow the user to modify the system generated code.

**REQ-7: Automate Reservation Email**: When a new user is created, the information relating to the reservation should be emailed out.

**REQ-8: Create Unique Credentials:** All Hotel Tango users need to have the ability to have their own user accounts.

## **System Sequence Diagrams**

### **Create Customer Sequence Diagram**

Diagram

Description automatically generated

### **Create Room Type Sequence Diagram**

Diagram

Description automatically generated

### **Create Room Sequence Diagram**

Diagram

Description automatically generated

### **Create Reservation Sequence Diagram**

Diagram

Description automatically generated with low confidence

# **User Effort Estimation using Use Case Points**

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Least Amount of Clicks** | **Explanation** |
| Login Screen | 2 | Here the user will click a login button and enter their username and password |
| Create a customer | 9 | When the customer clicks the nav bar to reach the customer page. They will enter all attributes for a customer and click create and create a customer via the createCustomer method |
| Modify a customer | 3 | On the view page for a customer, there will be a button to press to edit said customer, the user enters the changes that want and clicks an update button, which will then update the database via the updateCustomer method. |
| Create a room type | 5 | The user will click room type from the nav bar and click create room type from the room types view screen. Here they will enter all attributes and click create. The record will be created from the createRoomType method. |
| Modify a room type | 2 | On the view page for a room type, there will be a button to press to edit said room type. The user enters the changes that want and clicks an update button, which will then update the database via the updateRoomType method. |
| Create a room | 2 | The user will click room from the nav bar and click create room from the room view screen. Here they will enter all attributes and click create. The record will be created from the createRoom method. |
| Modify a room | 2 | On the view page for a room, there will be a button to press to edit said room. The user enters the changes that want and clicks an update button, which will then update the database via the updateRoom method. |
| Create a reservation | 6 | The user will click reservations from the nav bar and click create reservation. For subclasses such as customer and room, there will be a dropdown box to choose these values. Once complete, the user will click create and the database will be updated via the createCustomer class and the createWIFI Class. |
| Modify a reservation | 2 | The user will click reservations from the nav bar and click modify reservation. The subclass fields will be populated dropdown menus. Once the requested changes have been made, the user will click update and the datababase will be updated via the modifyReservation method. |
| Email a reservation | 1 | I haven’t figured out how this will work yet, but I imagine we will have a custom function that updates the database. |
| Modify WI-FI | 2 | This will be able to be updated from the modify reservation window. |

# **Domain Analysis**

Diagram

Description automatically generated with low confidence

# **Interaction Diagrams**

## **Customer**

Once an employee enters the data on the web form, their data will be passed into the createCustomer, updateCustomer, or DeleteCustomer method, then we return the data via the GetCustomer method.

Diagram

Description automatically generated

## **RoomType**

Once an employee enters the data on the web form, their data will be passed into the createRoomType, updateRoomType, or DeleteRoomType method, then we return the data via the GetRoomType method.

Diagram

Description automatically generated

## **Room**

Once an employee enters the data on the web form, their data will be passed into the createRoomType, updateRoomType, or DeleteRoomType method, then we return the data via the GetRoomType method.

Diagram

Description automatically generated

## **Reservation**

### **Create Reservation**

When creating a reservation, we depend on a few dependent classes. We get the available CustomerID’s, Room Id’s and RoomTypes. We then take that data and pass it into the CreateReservation method, where we also generate a random WIFI string from the database.

Diagram

Description automatically generated

### **Update Reservation**

When the employee chooses the reservation, they wish to modify, we pass that data into the UpdateReservation method and update the database.

Diagram

Description automatically generated

### **Get Reservation**

When the employee wishes to view a reservation, we pass the ID into the GetReservation method and return all applicable data.

Diagram

Description automatically generated

### 

### **Delete Reservation**

When the employee chooses the reservation, they wish to delete, we pass that data into the DeleteReservation method and update the database.

Diagram

Description automatically generated

# **Class Diagram and Interface Specification**

## **Class Diagram**

A picture containing timeline

Description automatically generated

## **Traceability Matrix**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Req’t | PW | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 | UC7 |
| REQ1 | 5 |  | x | x | x | x | x | x |
| REQ2 | 3 |  |  |  |  |  |  |  |
| REQ3 | 4 |  | x |  |  |  | x |  |
| REQ4 | 5 |  |  | x |  |  |  |  |
| REQ5 | 5 |  |  |  |  |  |  |  |
| REQ6 | 2 |  | x | x | x | x | x | x |
| REQ7 | 2 |  | x |  |  |  |  |  |
| REQ8 | 2 | x |  |  |  |  |  |  |
| Max PW | | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Total PW | | 0 | 11 | 12 | 7 | 7 | 11 | 7 |

# **System Architecture and System Design**

## **Architectural Structure**

The architectural structure of my application is the Model-View-Controller. The database is derived from classes. We then have views that are displayed to the user which displays data from the various classes. Then, we use controllers to pass data to and from the database.

## **Mapping Subsystems to Hardware**

This system will run on a single server. In a real-world scenario, this would run on a server from a cloud provider. This would allow the ability to scale the system up and out. As we scale resources appropriately, this would allow us to not have the need for multiple machines. We can have client machines that can access a single end point.

## **Persistent Data Storage**

We are using a relational database that will be geo-replicated. As the user actions are taken in the system and changes are made, these changes will be asynchronously committed to a relational database. Also, due to our relational database having ACID compliance, we can guarantee that transactions will be persisted and written correctly to the database.

## **Network Protocol**

This app will run on one machine, but all application traffic will converse via GET/PUT API, HTTPS and SQL Server ODBC.

# **Algorithms and Data Structures**

## **Algorithms**

Since my system is based on creating hotel reservations there were no algorithms implemented.

## **Data Structure**

My system does not have any complex data structures. At best, we pass variables of different data types to various classes to modify data and/or return data to the UI for end users.

# **User Interface Design and Implementation**

My initial screen mock-ups have stayed the same. The reason being, is I have not deviated much from my original plan. The only thing that has really changed is the search boxes and filtering feature.

# **Design of Tests**

I designed my tests around the database functionality, I had some trouble figuring out how to properly design unit testing. I tested my constraints and cascading deletes.

# **History of Work, Current Status, and Future Work**

This is the final part of our project. I was able to achieve everything that I wanted to achieve, except for creating proper unit tests. At this point, I am considering my project to be finished with no future work ahead.

# **References**

[Getting Started - EF Core | Microsoft Learn](https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli)

[Tutorial: Add sorting, filtering, and paging with the Entity Framework in an ASP.NET MVC application | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/sorting-filtering-and-paging-with-the-entity-framework-in-an-asp-net-mvc-application)

[Code First Approach In ASP.NET Core MVC With EF Core Migration (c-sharpcorner.com)](https://www.c-sharpcorner.com/article/code-first-approach-in-asp-net-core-mvc-with-ef-core-migration/)