**FIT Company**

**Carti Hotel Management System**

**Software Architecture Document**

**Version 1**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 20/11/2024 | 1.0 | Final version | Phan Khanh Huyen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1. Introduction 4](#_heading=h.30j0zll)

[2. Architectural Representation 4](#_heading=h.tyjcwt)

[3. Architectural Goals and Constraints 4](#_heading=h.1t3h5sf)

[4. Use-Case View 4](#_heading=h.2s8eyo1)

[4.1.3. Book rooms 6](#_heading=h.26in1rg)

[4.1.4. Make payment 6](#_heading=h.lnxbz9)

[4.1.5. Check-in 6](#_heading=h.35nkun2)

[4.1.6. Check-out 6](#_heading=h.1ksv4uv)

[4.1.7. Manage Services 7](#_heading=h.44sinio)

[4.1.8. Manage Users 7](#_heading=h.2jxsxqh)

[4.1.9. Manage Rooms 7](#_heading=h.z337ya)

[4.1.10. Display Revenue Report 7](#_heading=h.3j2qqm3)

[5. Logical View 7](#_heading=h.4i7ojhp)

[Class Receptionist: 10](#_heading=h.3whwml4)

[Class Reservation: 12](#_heading=h.2bn6wsx)

[Class Bill: 14](#_heading=h.qsh70q)

[Class Room: 15](#_heading=h.3as4poj)

[Class LoginController: 17](#_heading=h.1pxezwc)

[Class LogoutController: 18](#_heading=h.49x2ik5)

[Class RoomController: 19](#_heading=h.2p2csry)

[Class CustomerController: 21](#_heading=h.147n2zr)

[Class ReservationController: 23](#_heading=h.3o7alnk)

[Class ServiceController: 24](#_heading=h.23ckvvd)

[6. Process View 26](#_heading=h.32hioqz)

[7. Deployment View 27](#_heading=h.41mghml)

[8. Size and Performance 28](#_heading=h.2grqrue)

[9. Quality 29](#_heading=h.3fwokq0)

**Software Architecture Document**

1. **Introduction**
   1. **Purpose**

This document provides a comprehensive architectural overview of the Carti Hotel Management System, detailing various architectural aspects and key decisions made during its development. It serves to encapsulate and communicate the system's critical architectural considerations.

* 1. **Scope**

This document applies to the Carti Hotel Management System.

* 1. **Definitions, Acronyms, and Abbreviations**

**User**: Any individual interacting with the system, including customers, receptionists, and administrators.

**Receptionist**: A user responsible for managing room reservations and assisting customers.

**Administrator**: A user in charge of overseeing personnel, hotel facilities, and revenue management.

**Customer**: A user seeking to reserve rooms and access hotel services.

* 1. **References**

None.

* 1. **Overview**

The following sections provide a detailed exploration of the architectural design of the Hotel Management System. They outline the primary software architecture and examine the goals and constraints influencing the product’s quality, including security, distribution, and reusability. Key system views are presented to highlight various perspectives, while criteria related to system size, performance, and quality are proposed.

1. **Architectural Representation**

This document leverages Visual Paradigm Community Edition Models, StarUML, and the Unified Modeling Language (UML) to depict critical perspectives, including the Use-Case View, Logical View, Deployment View, and Data View.

1. **Architectural Goals and Constraints**

Several critical requirements and constraints shape the architecture of the system:

**Requirement Compliance**: The Carti Hotel Management System must meet all defined system requirements.

**Robustness and Adaptability**: The design should be resilient and capable of adapting to evolving functional requirements.

**Business Logic Reusability**: Reuse of business logic across applications is essential, requiring the clear separation of model, view, and controller components.

**Team Collaboration**: This separation supports efficient collaboration among various development teams.

**Platform and Database Access**: The system will operate on a dedicated platform with access to a database.

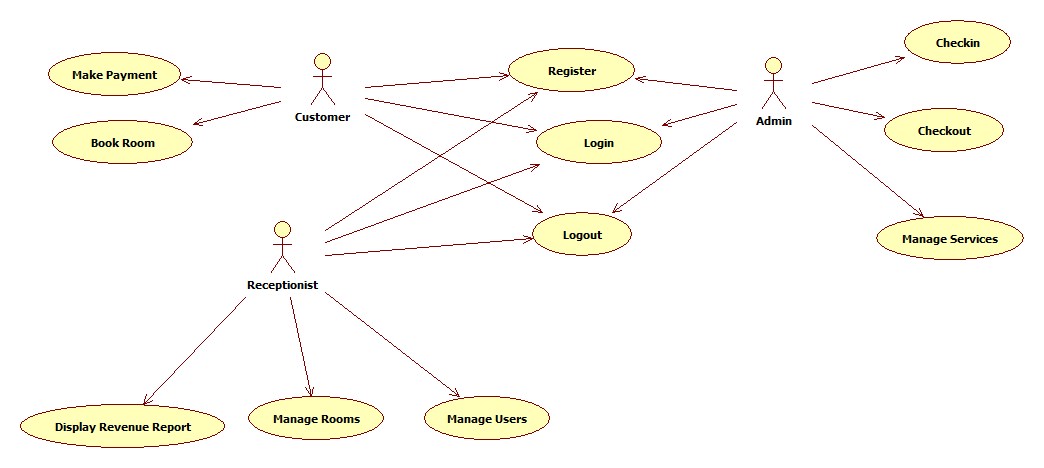
**Traffic Handling**: The website will manage most content display, so the system interface must handle high volumes of traffic effectively.

1. **Use-Case View**

The system is designed using the Model-View-Controller (MVC) architecture, ensuring consistent architectural coverage for all use cases. Among the system's use cases (as depicted in the diagram below), the most critical ones, outlined in the Requirement Definition folder, include:

* **Customer Role**: Book Rooms, Make a Payment
* **Receptionist Role**: Manage Services, Check-In, Check-Out

**Administrator Role**: Manage Users, Manage Rooms, Display Revenue Report



* 1. **Use- case Realizations**

This section elucidates the functioning of the software by presenting key use-case realizations. Detailed specifications for Use-Case Realizations can be found in the corresponding folder.

* + 1. **Login**

Brief Description: This use case outlines the process of a user logging into the Hotel Management System. The actors initiating this use case are the Customer, Receptionist, and Administrator.

* + 1. **Register**

Brief Description: This use case illustrates how a user registers a new account in the Hotel Management System. The actor initiating this use case is the Customer.

* + 1. **Book rooms**

Brief Description: Customers can use this use case to reserve rooms offered by the hotel. The actor in this use case is the Customer.

* + 1. **Make payment**

Brief Description: This use case allows customers to prepay for previously reserved rooms through online banking or credit cards. The actor in this use case is the Customer.

* + 1. **Check-in**

Brief Description: Receptionists can use this use case to make a direct reservation for customers at the reception, especially if the customer has not reserved a room through the application. The actor in this use case is the Receptionist.

* + 1. **Check-out**

Brief Description: This use case allows receptionists to check-out for customers. The actor of this use case is Receptionist

Brief Description: This use case enables receptionists to check out customers. The actor in this use case is the Receptionist.

* + 1. **Manage Services**

Brief Description: This use case allows receptionists to manage services, including adding services if the customer wants to order any services offered by the hotel and deleting services. The actor of this use case is Receptionist

* + 1. **Manage Users**

Brief Description: This use case allows administrators to manage users, including adding a new user (customer/ receptionist), updating an existing user information and delete an existing user from the database. The actor of this use case is Administrator.

* + 1. **Manage Rooms**

Brief Description: This use case allows administrators to manage rooms, including adding a new room, updating an existing room information and deleting an existing user from the database. The actor of this use case is Administrator.

* + 1. **Display Revenue Report**

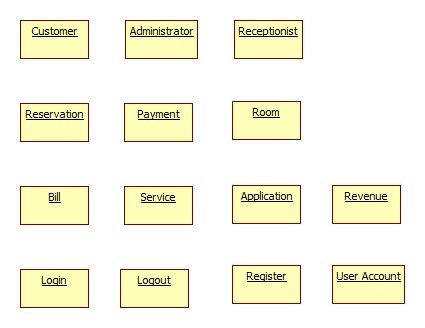
Brief Description: This use case allows administrators to see the daily, weekly, monthly, annual revenue of the hotel (income, loss in terms of money). The actor of this use case is Administrator.

1. **Logical View**
   1. **Overview**

The architectural perspective focuses on the logical structure of the system, breaking down the design model into a hierarchical package structure with distinct layers. The logical view of the Hotel Management System consists of three main packages:

* **Model Package**: Contains entity objects that represent core business elements and tangible artifacts, reflecting the system's data and business logic.
* **View Package**: Includes classes responsible for generating output representations of information for users, based on changes in the model. These classes handle the presentation layer.
* **Controller Package**: Comprises classes that control the interactions between the model and the view. These classes can issue commands to the model to modify its state (e.g., adding a new room) and adjust how the model's data is presented to the user in the view (e.g., updating room listings).
  1. **Architecturally Significant Design Packages**
     1. Models

The diagram below describes all model classes in the Models Package:



In the Hotel Management System, the core functionality is driven by key classes, each responsible for specific aspects of the system. These pivotal classes include:

* **Room**: Manages information related to hotel rooms, such as availability, pricing, and types.
* **Customer**: Handles customer details, including personal information and booking history.
* **Receptionist**: Manages customer interactions at the front desk, including reservations and check-ins/outs.
* **Administrator**: Oversees the system’s operation, managing users, rooms, and revenue.
* **Reservation**: Manages room booking details, including dates, customers, and room assignments.
* **Revenue**: Tracks financial data, including hotel income and losses.
* **Service**: Manages additional services offered by the hotel, such as room service or amenities.
* **Bill**: Generates and maintains customer billing information, including room charges and additional services.
* **Payment**: Handles payment processing for bookings and services.
* **UserAccount**: Manages user login credentials and access rights within the system.

These classes interact to form the backbone of the system, enabling efficient management of the hotel's operations and customer interactions.

**Class UserAccount:**

|  |  |
| --- | --- |
| **Name** | UserAccount |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| accountId | long | Private | False | False | N/A | 1 | N/A |
| username | String | Private | True | False | 30 | N/A | N/A |
| email | String | Private | True | False | 30 | N/A | N/A |
| password | String | Private | True | False | 20 | N/A | N/A |
| createdAt | Date | Private | False | False | N/A | N/A | N/A |
| phone | String | Private | True | False | 15 | N/A | N/A |
| roleId | long | Private | False | False | N/A | 1 | 3 |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the user account to the database. Return true if success | | | |
| update() | boolean | Public | Instance | Update the user account in the database. Return true if success | | | |
| delete() | boolean | Public | Instance | Delete the user account from the database. Return true if success | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |
| getAllUserAccounts() | List | Public | Classifier | Return all user accounts in database as a list | | | |
| getUserAccount | UserAcc ount | Public | Classifier | Return a user account in database with specified identifier | | | |

**Class Customer:**

|  |  |
| --- | --- |
| **Name** | Customer |
| **Brief Description** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| customerId | long | Private | False | False | N/A | 1 | N/A |
| name | String | Private | True | False | 30 | N/A | N/A |
| phone | String | Private | True | False | 15 | N/A | N/A |
| dateOfBirth | Date | Private | True | False | N/A | N/A | N/A |
| nationality | String | Private | False | False | 20 | N/A | N/A |
| passportNumber | String | Private | False | True | 20 | N/A | N/A |
| balance | double | Private | True | False | N/A | 0 | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the customer in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the customer information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the customer from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |
| getAllCustomer() | List | Public | Classifier | Return all customers in the database as a list | | | |
| getCustomer | Customer | Public | Classifier | Return a customer in the database with specified identifier | | | |

**Class Receptionist:**

|  |  |
| --- | --- |
| **Name** | Receptionist |
| **Brief Description** |  |
| **Attributes** | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| receptionistId | long | Private | False | False | N/A | 1 | N/A |
| name | String | Private | True | False | 20 | N/A | N/A |
| email | String | Private | True | False | 30 | N/A | N/A |
| dateHired | Date | Private | False | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the receptionist in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the receptionist information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the receptionist from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |
| getAllReceptionists() | List | Public | Classifier | Return all receptionists in database as a list | | | |
| getReceptionist | Reception ist | Public | Classifier | Return a receptionist in database with specified identifier | | | |

**Class Administrator:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | Administrator | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| adminId | long | Private | False | False | N/A | 1 | N/A |
| name | String | Private | True | False | 20 | N/A | N/A |
| email | String | Private | True | False | 30 | N/A | N/A |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| phoneNumber | String | Private | True | False | 15 | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the administrator in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the administrator information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the administrator from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |
| getAdministrator | Reception ist | Public | Classifier | Return an administrator in database with specified identifier | | | |

**Class Reservation:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | Reservation | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| reservationId | long | Private | False | False | N/A | 1 | N/A |
| customerId | long | Private | False | False | N/A | 1 | N/A |
| roomId | long | Private | False | False | N/A | 1 | N/A |
| dateStart | Date | Private | True | False | N/A | N/A | N/A |
| dateEnd | Date | Private | True | False | N/A | N/A | N/A |
| numberOfPeople | int | Private | True | False | N/A | 1 | 4 |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| add() | boolean | Public | Instance | Add the reservation in the database and return a Boolean value indicating success (true if successful). |
| update() | boolean | Public | Instance | Modify the reservation information in the database and return true if the operation is successful. |
| delete() | boolean | Public | Instance | Remove the reservation from the database and return true if the operation is successful. |
| save() | boolean | Private | Instance | Save change from this to database |
| getReservation() | Reservati on | Public | Classifier | Return a reservation in the database with specified identifier |

**Class Service:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | Service | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| serviceId | long | Private | False | False | N/A | 1 | N/A |
| reservationId | long | Private | False | False | N/A | 1 | N/A |
| type | String | Private | True | False | 30 | N/A | N/A |
| quantity | int | Private | True | False | N/A | 1 | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the service in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the service information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the service from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| getService() | Service | Public | Classifier | Return service in database with specified identifier |

**Class Bill:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | Bill | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| billId | long | Private | False | False | N/A | 1 | N/A |
| reservationId | long | Private | False | False | N/A | 1 | N/A |
| totalPrice | double | Private | True | False | N/A | 1.0 | N/A |
| servicesOrdered | List | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the bill in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the bill information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the bill from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |
| getBill() | Bill | Public | Classifier | Return a bill in the database with specified identifier | | | |

**Class Payment:**

|  |  |
| --- | --- |
| **Name** | Payment |
| **Brief Description** |  |
| **Attributes** | |

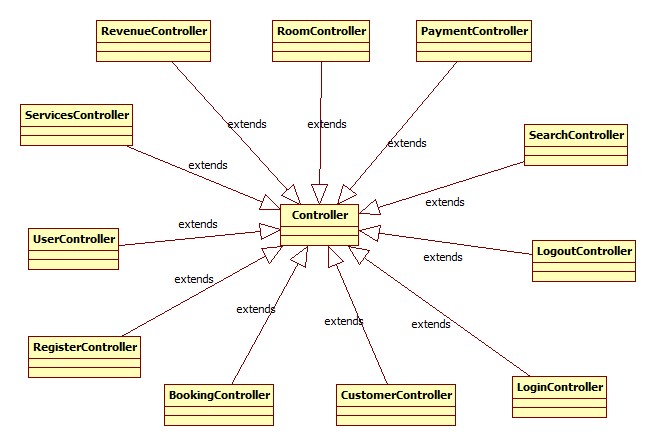
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| paymentId | long | Private | False | False | N/A | 1 | N/A |
| reservationId | long | Private | False | False | N/A | 1 | N/A |
| totalPrice | double | Private | True | False | N/A | 1.0 | N/A |
| type | String | Private | False | False | 10 | N/A | N/A |
| status | String | Private | True | False | 10 | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the payment in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the payment information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the payment from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save change from this to database | | | |
| getPayment() | Payment | Public | Classifier | Return a payment in the database with specified identifier | | | |

**Class Room:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | Room | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| roomId | long | Private | False | False | N/A | 1 | N/A |
| roomNumber | String | Private | True | False | 4 | N/A | N/A |
| name | String | Private | True | False | 20 | N/A | N/A |
| peopleCapacity | int | Private | True | False | N/A | 1 | 4 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| desc | String | Private | True | True | 100 | N/A | N/A |
| pricePerNight | double | Private | True | False | N/A | 1.0 | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| add() | boolean | Public | Instance | Add the room in the database and return a Boolean value indicating success (true if successful). | | | |
| update() | boolean | Public | Instance | Modify the room information in the database and return true if the operation is successful. | | | |
| delete() | boolean | Public | Instance | Remove the room from the database and return true if the operation is successful. | | | |
| save() | boolean | Private | Instance | Save changes from this to the database. | | | |
| getAllRooms() | List | Public | Classifier | Retrieve all rooms in the database as a list. | | | |
| getRoom() | Payment | Public | Classifier | Retrieve a room in the database with the specified identifier. | | | |

* + 1. **Controllers**



**Class LoginController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | LoginController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| username | String | Private | True | False | 20 | N/A | N/A |
| password | String | Private | True | False | 20 | N/A | N/A |
| isLoggedIn | boolean | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** |
| validate() | boolean | Public | Instance | Verify the username and password, returning true if the credentials are valid. |
| login() | void | Public | Instance | Enable the user to log into the system. |
| requestHomePage() | void | Public | Instance | Request the Home Page View associated with the user's role. |
| displayError() | void | Public | Instance | Handle requests for displaying errors. |

**Class RegisterController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | RegisterController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| username | String | Private | True | False | 20 | N/A | N/A |
| email | String | Private | True | False | 30 | N/A | N/A |
| password | String | Private | True | False | 20 | N/A | N/A |
| confirmPassword | String | Private | True | False | 20 | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| checkRegisterInformat ion() | boolean | Public | Instance | Verify the username and password, returning true if the credentials are valid. | | | |
| checkUserInfo() | boolean | Public | Instance | Return true if the user is present in the database. | | | |
| registerUser() | void | Public | Instance | Add the user to the database if the credentials are valid | | | |
| displayError() | void | Public | Instance | Handle requests for displaying errors. | | | |

**Class LogoutController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | LogoutController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| isLoggedIn | boolean | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| endCurrentSession() | void | Public | Instance | Terminate the current user session. | | | |
| confirm() | boolean | Public | Instance | Handle the confirmation of a specific action. | | | |

**Class SearchController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | SearchController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| searchKeyWord() | String | Private | True | False | 50 | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| searchCustomerName() | List | Public | Instance | Handling search customers request. | | | |
| searchRoom() | List | Public | Instance | Handling search rooms request | | | |

**Class RoomController:**

|  |  |
| --- | --- |
| **Name** | RoomController |
| **Brief Description** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| room | Room | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| showRoomList() | List | Public | Instance | Manage the request to display rooms. | | | |
| loadRoom() | void | Public | Instance | Handle requests to load rooms, including room number and availability status. | | | |
| loadRoomsInfo() | void | Public | Instance | Handle requests to load rooms, including all information related to the rooms. | | | |
| loadRoomInfo() | void | Public | Instance | Handle requests to load a single room. | | | |
| checkRoomExist() | boolean | Public | Instance | Return true if the room exists in the database. | | | |
| checkConfirmed() | boolean | Public | Instance | Return true if the user confirms a specific action. | | | |
| getListOfRooms() | List | Public | Instance | Handle requests to get all rooms. | | | |
| getListOfProperRoom () | List | Public | Instance | Handle requests to get a list of matching rooms. | | | |
| getRoomInfo() | Room | Public | Instance | Handle requests to get a single room. | | | |
| loadRoomImages() | void | Public | Instance | Handle requests to load room images of the most featured, available room. | | | |
| showRoomInfo() | void | Public | Instance | Handle requests to show a room. | | | |
| addRoom() | boolean | Public | Instance | Handle requests to add a room and return true if successful. | | | |
| updateRoom() | boolean | Public | Instance | Handle requests to update a room and return true if successful. | | | |
| deleteRoom() | boolean | Public | Instance | Handle requests to delete a room and return true if successful. | | | |

**Class UserController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | UserController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| user | User | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| displayUsers() | List | Public | Instance | Manage the request to display all users. | | | |
| checkUserExist() | boolean | Public | Instance | Return true if the user exists in the database. | | | |
| validateInformation() | boolean | Public | Instance | Validate all information for adding a new user and return true if all information is valid. | | | |
| loadUsers() | void | Public | Instance | Handle requests to load all users. | | | |
| getListOfUsers() | List | Public | Instance | Handle requests to get all users. | | | |
| getUserInfo() | User | Public | Instance | Handle requests to get a single user. | | | |
| addUser() | boolean | Public | Instance | Handling add user request. Return true if success | | | |
| updateUser() | boolean | Public | Instance | Handle requests to add a user and return true if successful. | | | |
| deleteUser() | boolean | Public | Instance | Handle requests to update a user and return true if successful. | | | |
| checkConfirmed() | boolean | Public | Instance | Return true if the user confirms a specific action. | | | |

**Class CustomerController:**

|  |  |
| --- | --- |
| **Name** | CustomerController |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| customer | Customer | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| showCustomers() | List | Public | Instance | Manage the request to display all customers. | | | |
| loadCustomers() | void | Public | Instance | Handle requests to load all customers. | | | |
| getListOfCustomers() | List | Public | Instance | Handle requests to get all customers.  . | | | |
| getCustomerById() | Customer | Public | Instance | Handle requests to get a single customer | | | |

**Class BookingController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | BookingController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| booking | Booking | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| updateRoomsAvailabil ity() | void | Public | Instance | Handling update room’s availability request. Return true if success | | | |
| updateReservation() | boolean | Public | Instance | Handling update reservation request. Return true if success | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| saveCheckInInfor() | void | Public | Instance | Manage the request to save check-in information. |
| loadBill() | void | Public | Instance | Handle requests to load a bill. |
| getBill() | Bill | Public | Instance | Handle requests to get the bill of a reservation. |
| showBill() | String | Public | Instance | Handle requests to show bill information. |
| printBill() | String | Public | Instance | Handle requests to print a bill. |
| checkOut() | void | Public | Instance | Handle check-out requests. |

**Class ReservationController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | ReservationController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| reservation | Reservati on | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| loadReservationById() | void | Public | Instance | Handle requests to load a single reservation. | | | |
| getReservationHistory () | List | Public | Instance | Handle requests to load all reservations of a customer. | | | |
| checkBooked() | boolean | Public | Instance | Return true if the customer has reserved any room. | | | |
| updateReservation() | boolean | Public | Instance | Handle requests to update a reservation and return true if successful. | | | |

**Class PaymentController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | PaymentController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| payment | Payment | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| makePayment() | void | Public | Instance | Handling make payment request | | | |
| returnPaymentResult() | String | Public | Instance | Handling return payment result from bank page request | | | |
| updatePayment() | boolean | Public | Instance | Handling update payment request. Return true if success | | | |

**Class ServiceController:**

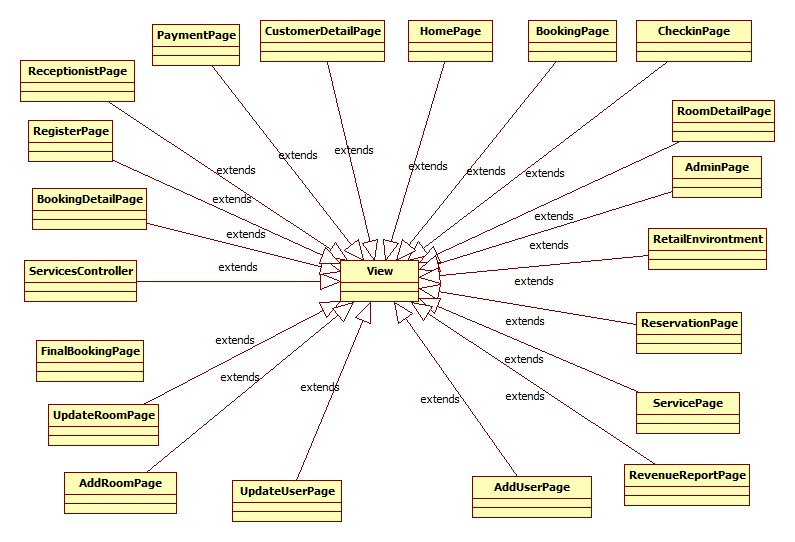
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | ServiceController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| service | Service | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| addService() | boolean | Public | Instance | Handling add a new type of service to the reservation request. Return true if success | | | |
| addServiceToBill() | boolean | Public | Instance | Handling add new services to a customer’s bill request. Return true if | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | success |
| deleteService() | boolean | Public | Instance | Handle requests to delete a type of service from the reservation and return true if successful. |
| deleteServiceInBill() | boolean | Public | Instance | Handle requests to delete services from a customer's bill and return true if successful. |
| getServiceInfo() | String | Public | Instance | Handle requests to get a service's information. |
| getAllServices() | List | Public | Instance | Handle requests to get all services of a reservation. |
| displayServices() | void | Public | Instance | Manage the request to display all services of a reservation. |

**Class ServiceController:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | | | | ServiceController | | | |
| **Brief Description** | | | |  | | | |
| **Attributes** | | | | | | | |
| **Name** | **Type** | **Access** | **Mutable** | **Optional** | **Length** | **Min** | **Max** |
| revenue | Revenue | Private | True | False | N/A | N/A | N/A |
| view | View | Private | True | False | N/A | N/A | N/A |
| **Operations** | | | | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** | | | |
| getRevenue() | Revenue | Public | Instance | Handle requests to get revenue. | | | |
| getRevenueInfo() | String | Public | Instance | Handle requests to get revenue information. | | | |
| showDailyRevenue() | void | Public | Instance | Handle requests to show daily revenue. | | | |
| showWeeklyRevenue() | void | Public | Instance | Handle requests to show weekly revenue. | | | |
| showMonthlyRevenue() | void | Public | Instance | Handle requests to show monthly revenue. | | | |
| showYearlyRevenue() | void | Public | Instance | Handle requests to show annual revenue. | | | |

* + 1. Views



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | View | | | |
| **Brief Description** | Represents the visualization of the data that model contains. | | | |
| **Implementing classes** | Retail Environment, LoginPage, HomePage, RegisterPage, CustomerPage, ReceptionistPage, AdminPage, BookingPage, RoomDetailPage, BookingDetailPage, FinalBookingPage, PaymentTypesPage, CheckinPage, ReservationDetailPage, ServicesPage, AddUserPage, UpdateUserPage, AddRoomPage, UpdateRoomPage, RevenueReportPage, BankPage | | | |
| **Operation** | | | | |
| **Header** | **Return type** | **Access** | **Scope** | **Specification** |
| setController(Controller) | void | Public | Instance | Map this view with the specified controller. |
| update() | void | Public | Instance | Update this view based on changes in model. |
| render() | void | Public | Instance | Render this view. |

1. **Process View**

The Carti Hotel Management System is intended for implementation on a Node.js server, which operates with a single-threaded asynchronous event handling model (event loop). Consequently, this document will not address concurrency issues.

1. **Deployment View**

This section outlines one or more physical network configurations where the Hotel Management System is deployed and executed. The system comprises essential physical nodes, including two firewalls (internal and external), a web server, a database server, and a backup database server. The diagram below provides a simplified representation of the Carti Hotel Management System deployment view.

1. **Size and Performance**

The primary dimensioning characteristics influencing the software's architecture and performance constraints are as follows:

1. User Capacity

- The software should support a maximum of 2000 concurrent users accessing the main database and 100 simultaneous users accessing local servers concurrently.

1. fficiency and Accuracy

- All tasks performed by the system must incur minimal delays, and the system should ensure accurate storage of information during transactions.

1. Legacy Database Integration

- The system needs to facilitate access to the legacy course catalog database with a latency not exceeding 5 seconds. This emphasizes the requirement for seamless integration with legacy systems.

1. Transaction Completion Time

- The system is expected to complete 80% of transactions in less than two minutes. This sets a performance standard for the system's ability to handle and process various tasks efficiently.

1. **Quality**

The system architecture supports the quality requirements:

* The system is capable of ensuring that all information transitions are saved in order to uphold the highest level of system integrity.
* Daily backups of databases will be performed out of concern for security issues.
* The system website can appear properly on many devices and web browsers with any screen size (i.e. responsive design).

Popular web browsers including Google Chrome, Mozilla Firefox, Opera, Safari, Microsoft Edge, and Internet Explorer provide access to all system website