Group Nice

Nice Hotel and Restaurant
Software Architecture Document

Version <2.0>

Revision History

Date	Version	Description	Author
28/11/2023	1.0	PA3 Version	Whole team
14/12/2023	2.0	PA4 Version	Whole team

Table of Contents

Software Architecture Document

1. Introduction

- This document aims to outline the architecture framework, design principles and structural components that govern the development and functionality of out website. This document serves as a comprehensive guide for out team involved in the creation, maintenance and enhancement of the website.
- The scope of this document encompasses the use-case model and logical view. And in this document, we will focus more on the logical view, which consists of components and helps set the stage for a better understanding of the entire system architecture.
- Acronym list:
  - HTTP: Hypertext Transfer Protocol.
- Reference:
  - MVC là gì? Ứng dung của mô hình MVC trong lập trình (vietnix.vn)

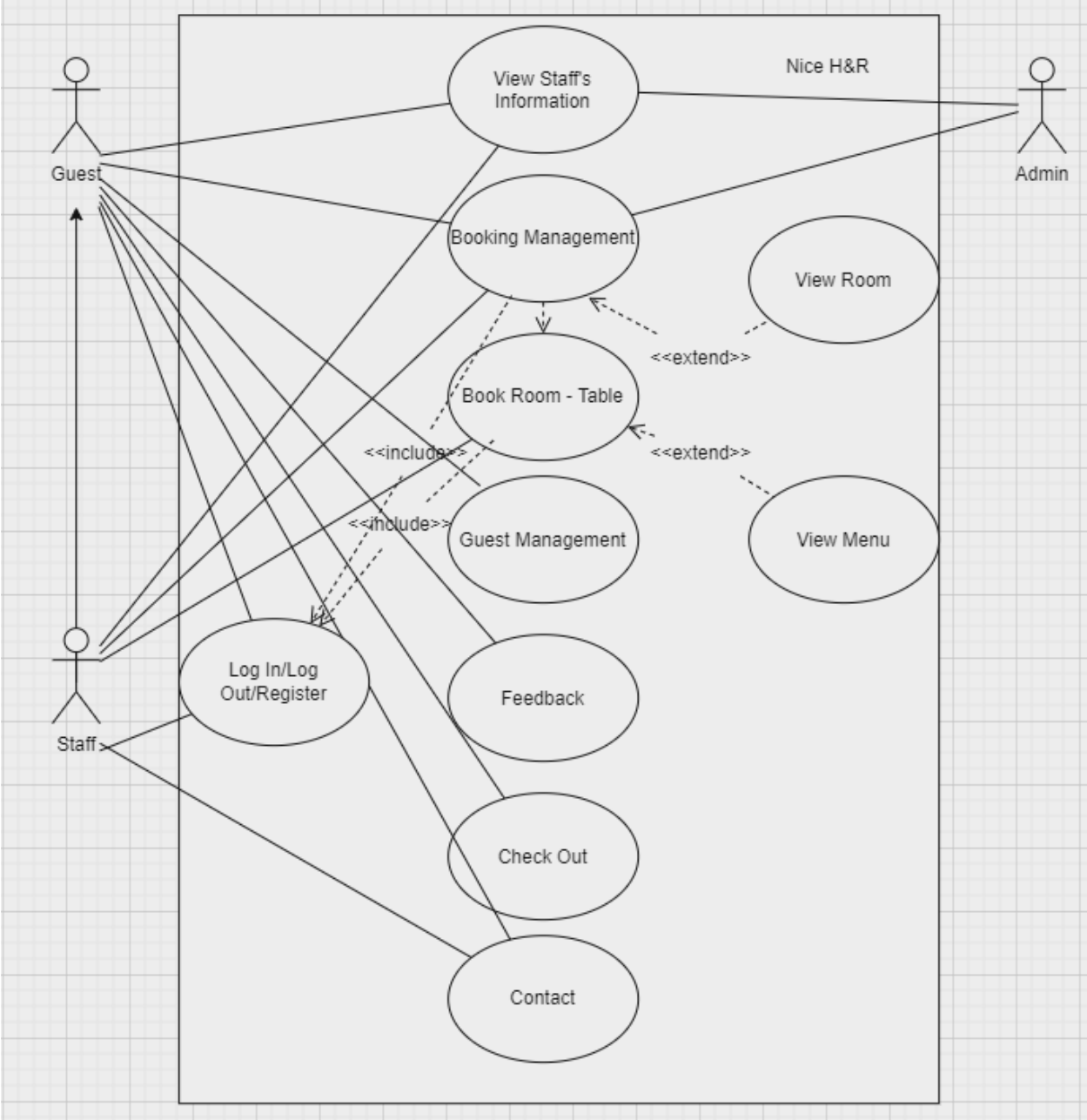
2. Architectural Goals and Constraints

- Technical requirements:
  - Programming language: HTML, CSS, JS
  - Framework: ReactJS
  - Database: MongoDB

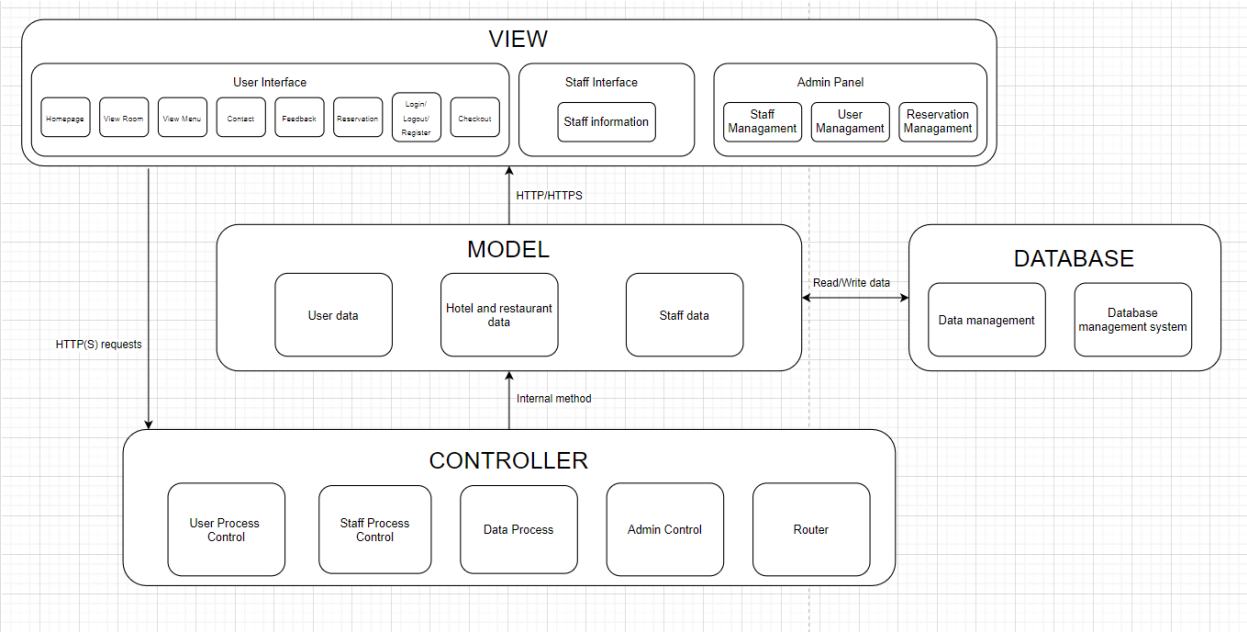
- Performance requirements:
  - Quick page load time
  - User-friendly interface

- Security requirements:
  - Use hash for password
  - Use gmail to log in

3. Use-Case Model



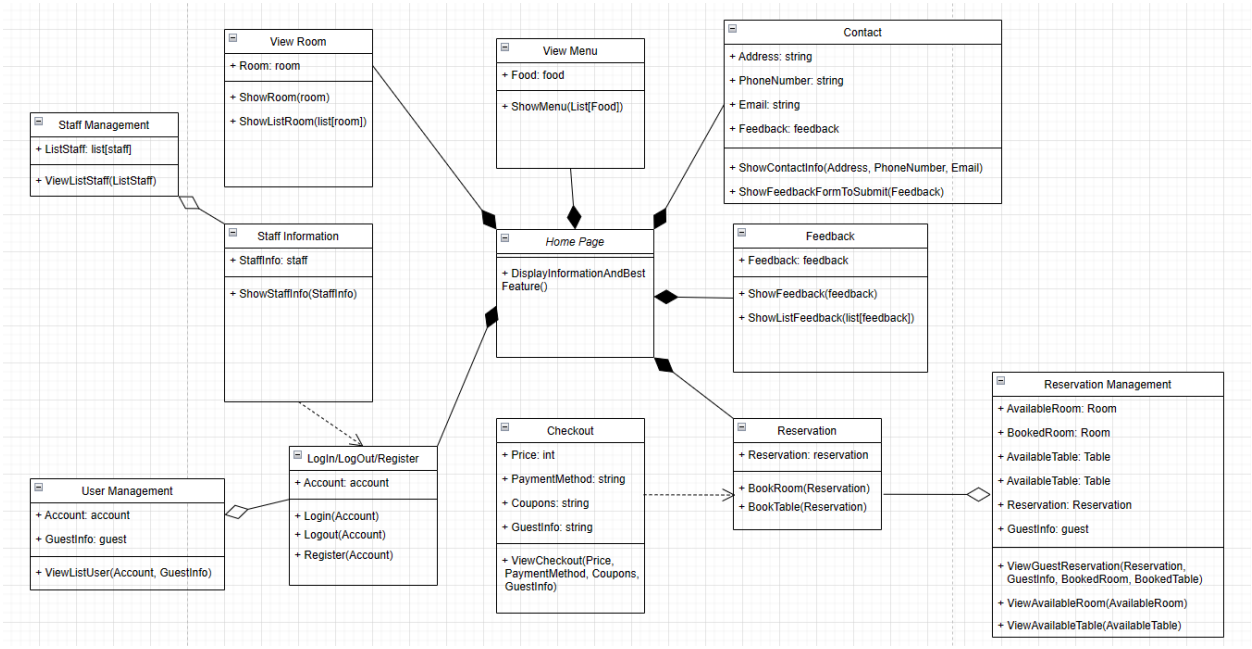
4. Logical View



❖ Model-View-Controller (MVC) pattern:

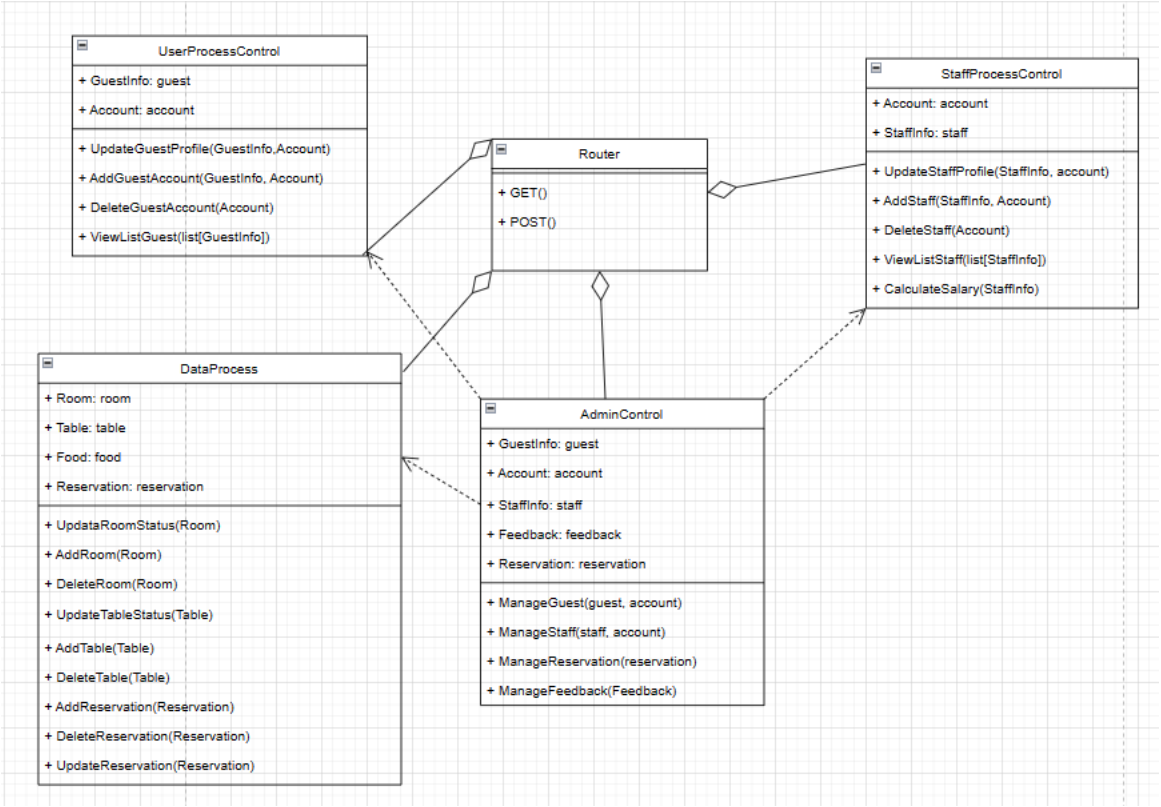
- The MVC pattern provides a structured approach to software development, promoting maintainability, scalability, and organization within applications by separating concerns and responsibilities among its three main components: Model, View, and Controller.
    - Model:
        - Represents the data and business logic of the application.
        - Handles data manipulation, storage, retrieval, and validation.
        - It doesn't directly interact with the user interface or presentation layer.
    - View:
        - Represents the user interface or the presentation layer.
        - Displays data from the Model to the users.
        - Responds to user input and sends requests to the Controller.
    - Controller:
        - Acts as an intermediary between the Model and the View.
        - Handles user input, processes requests from the View, and interacts with the Model accordingly.
        - Contains the application logic, determines which Model methods to call, and prepares data to send back to the View.
    - Database:
        - The database is responsible for storing, accessing, and updating data among many other functionalities. It plays a vital role in managing security and recovery services within the data management system, ensuring the enforcement of constraints in this underlying system.

4.1 Component: VIEW



- The View component is crucial for delivering an engaging and user-friendly interface to users. Its primary role is to present the data retrieved from the underlying Model to the users. It is responsible for structuring and formatting the information obtained from the Model but doesn't handle any business logic or data manipulation. It determines the visual representation of elements such as homepage, rooms, menu, contact infomation, feedback, login/logout/register, reservation, checkout, staff profiles, and other contents relevant to management.
    - Homepage: View some outstanding features of hotel and restaurant and our information.
    - Rooms: View room types and detailed of rooms such as price and amenities.
    - Menu: View menu and detailed of food with image.
    - Contact Information: View our contact information such as email and phone number and view feeback form for guest to give feedback after using our service
    - Feedback: View feedback of guests.
    - Login/Logout/Register: View form for guests to login, logout and register.
    - Reservation: View form that has information for guests to easily book room or table.
    - Checkout: View form that has information for guests to pay and checkout and show notification when successful checkout.
    - Staff profile: View staff profile such as full name, image, salary,...
    - Management: View information for staff and admin to manage staff and reservation.

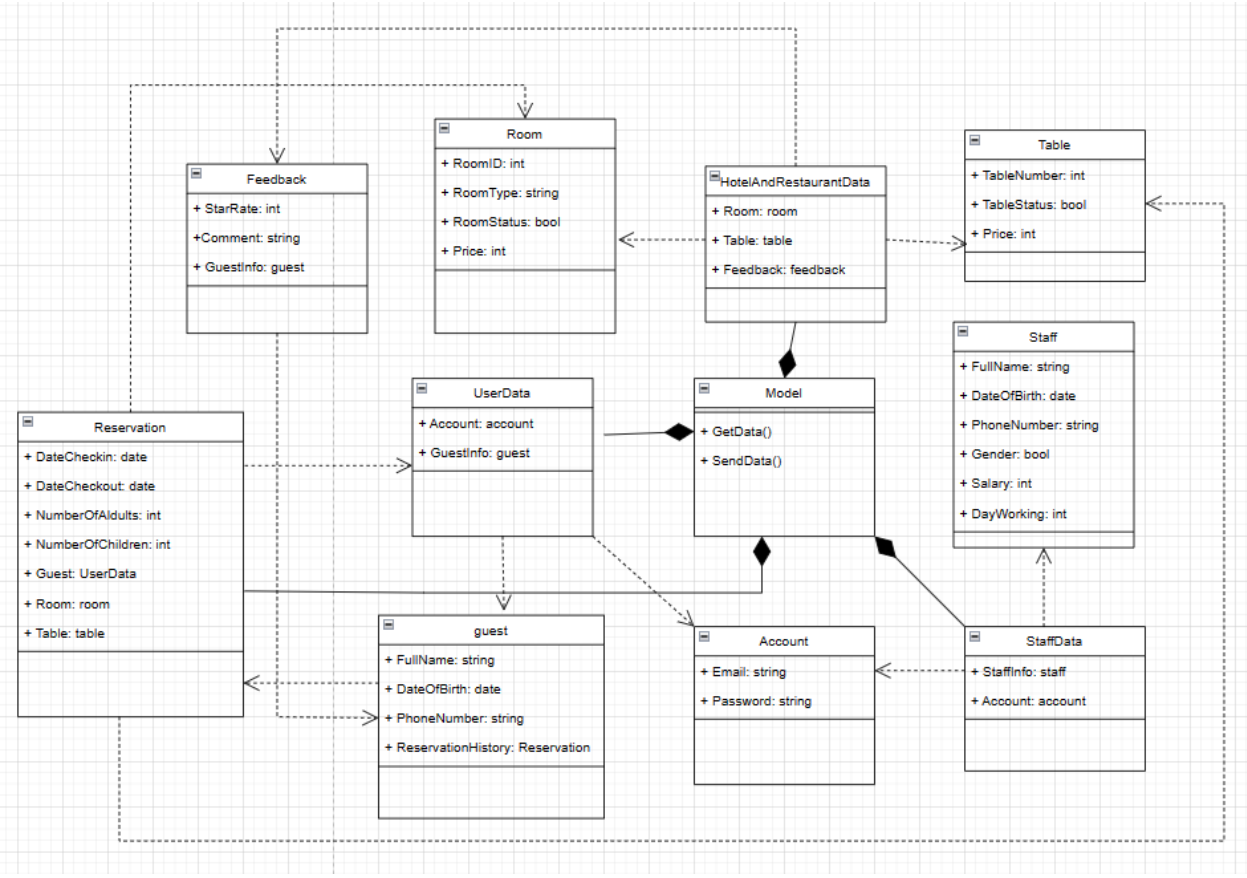
4.2 Component: CONTROLLER



The Controller component plays a pivotal role in managing user interactions, handling input, and orchestrating the flow of data between the View and the Model. Unlike the View, the Controller doesn't focus on the visual presentation or the specifics of the data itself; instead, it concentrates on controlling the application's behavior based on user actions. The Controller acts as an intermediary between the user interactions and the underlying data and business logic of the Model. It receives input from the user through the View, interprets those actions, and determines the appropriate responses. It does not contain detailed knowledge about the specific data or its structure but is responsible for handling user requests and delegating actions to the Model. The Controller coordinates the application's workflow, making decisions based on user interactions and triggering appropriate actions within the Model. It ensures that the user's requests are processed correctly, validating input, invoking necessary operations, and updating the Model accordingly. Moreover, the Controller manages the communication between the View and the Model, ensuring a clear separation of concerns and maintaining the integrity of the application's architecture.

- UserProcessControl: Control process relating to the users (guests) such as adding a new account
- StaffProcessControl: Control process relating to staff such as adding new staff, deleting staff
- DataProcess: Control process relating to data such as reservation
- AdminControl: Control 3 above processes and feedback.
- Router: forward the supported requests to the appropriate controller functions, get the request data from the Model and respond to View.

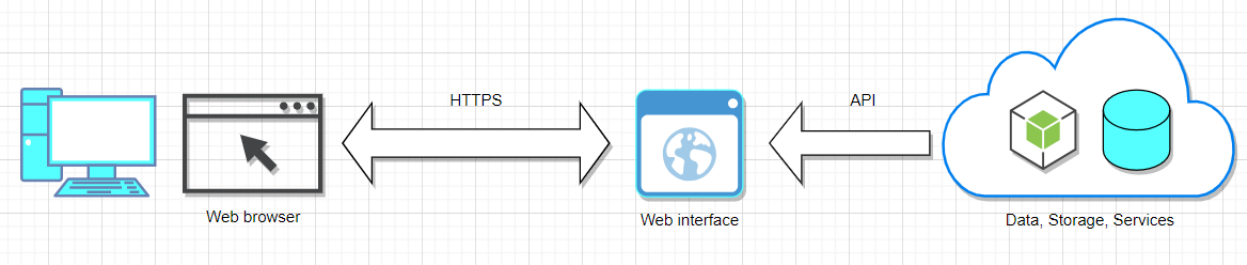
4.3 Component: MODEL



The Model component serves as the foundation of the application, responsible for managing the data, business logic, and core functionalities. It encapsulates the data-related operations and logic without being concerned with how the data is presented to users or how user interactions are handled. The Model encompasses aspects such as room, menu, reservation, staff, and other data relevant to the functioning of the website. It represents the underlying data structures, databases, and business rules that govern the behavior of the application. The Model's primary focus is on managing and manipulating the data, implementing business rules, performing validations, and ensuring data integrity. It doesn't interact directly with the user interface or manage how data is displayed to users; instead, it responds to requests from the Controller and provides the necessary data to fulfill those requests.

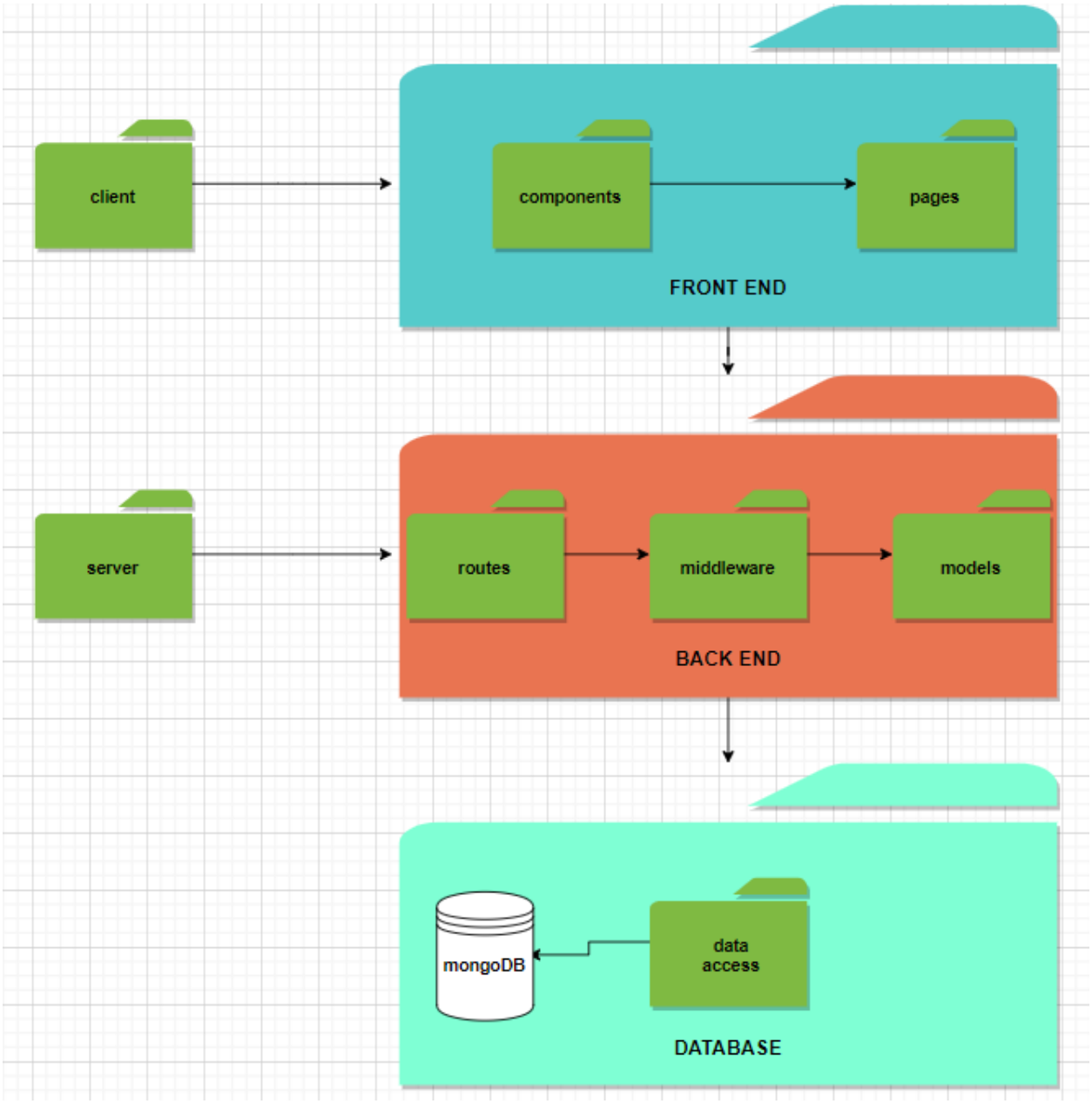
- Hotel and Restaurant Data: data relating to hotel and restaurant such as table and room.
- User Data: data relating to users (guests)
- Reservation: data used for reservation
- StaffData: data relating to staffs

5. Deployment



We strive to ensure that our users can seamlessly access the web application on personal computers (PCs) by optimizing compatibility with various browsers such as Chrome and Edge. Our focus is on creating responsiveness across different screen sizes and prioritizing optimal development for laptops and PCs. Our goal is to provide a smooth experience specifically tailored for PC users, while also considering compatibility with Windows, MacOS, and Linux operating systems.

6. Implementation View



We separate the **Front-end** folder into following subsections:

- The **components folder** includes several subfolders, each of which renders a specific component, such as Carousel, Dishes, GuestAndStaff, StuffDetail,…
- The **pages folder** contains several files, each of which defines the layout and components for a specific page.
- The **pages folder** can find necessary components in the **components.**

We separate the **Back-end** folder into following subsections:

- The **routes folder** includes necessary routers to handle the corresponding requests from users

- The **routes folder** can have access to the **middleware folder**, which defines necessary middleware functions to help verify user's information or the id of objects.
- The **routes folder** can modify the data stored in the database via the models defined in the **models folder.** There are 3 main models, which are user model, booking model and room model, each defines the necessary information to store in each model.

We store data in **database** system:

- The **MongoDB** database stores information about users, booking information and room information.