Hotel Management System: Ducks Nest

Software Design Specification

Author:

Luke Scribner, Julian Albert, Sophia Zhang, Cynthia Meneses Cervon, Reza Kamali

Table of Contents

1. System Overview

Using ReactJS, NextJS, and Firestore, the website aims to simplify various hotel management tasks while providing a seamless user experience for both hotel staff and guests.The system leverages React as the front-end framework to enable a dynamic and interactive user interface; NextJS is used for server-side rendering that will enhance performance. The integration with Firestore, a flexible NoSQL database, enables real-time data synchronization and efficient storage of hotel related information. The system's modular architecture and efficient design enable easy maintenance and offer a reliable tool for hotel administrators to optimize operations, and enhance guest experiences.

2. Software Architecture

2.1 Component Interface

2.2 Front End

Using ReactJS as a front-end framework can benefit the software architecture due to the component-based approach of React which allows for a modular and reusable code structure. In other words, different parts of the web app like header, booking form, and room listings can be built as separate components which makes it easier to develop, test, and maintain.

Additionally, React's virtual DOM (Document Object Model) can optimize performance since it updates and renders the necessary components efficiently. Consequently, React helps us to create an architecture that is organized, scalable, and efficient while ensuring a smooth user experience and facilitating future enhancements or modifications to the app.

See **Fig 1 & 2** for the front-end architecture

Fig 1. Client View

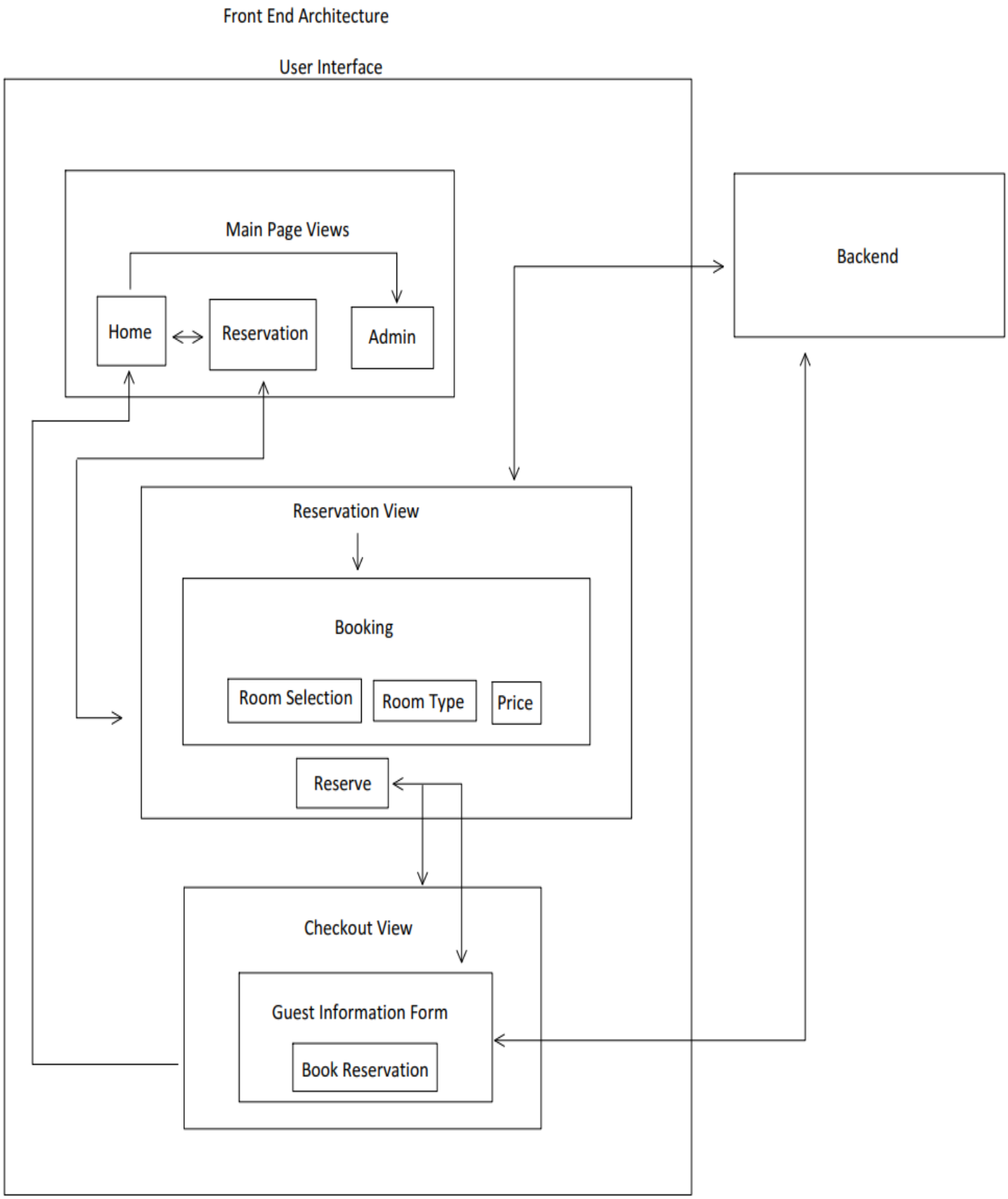
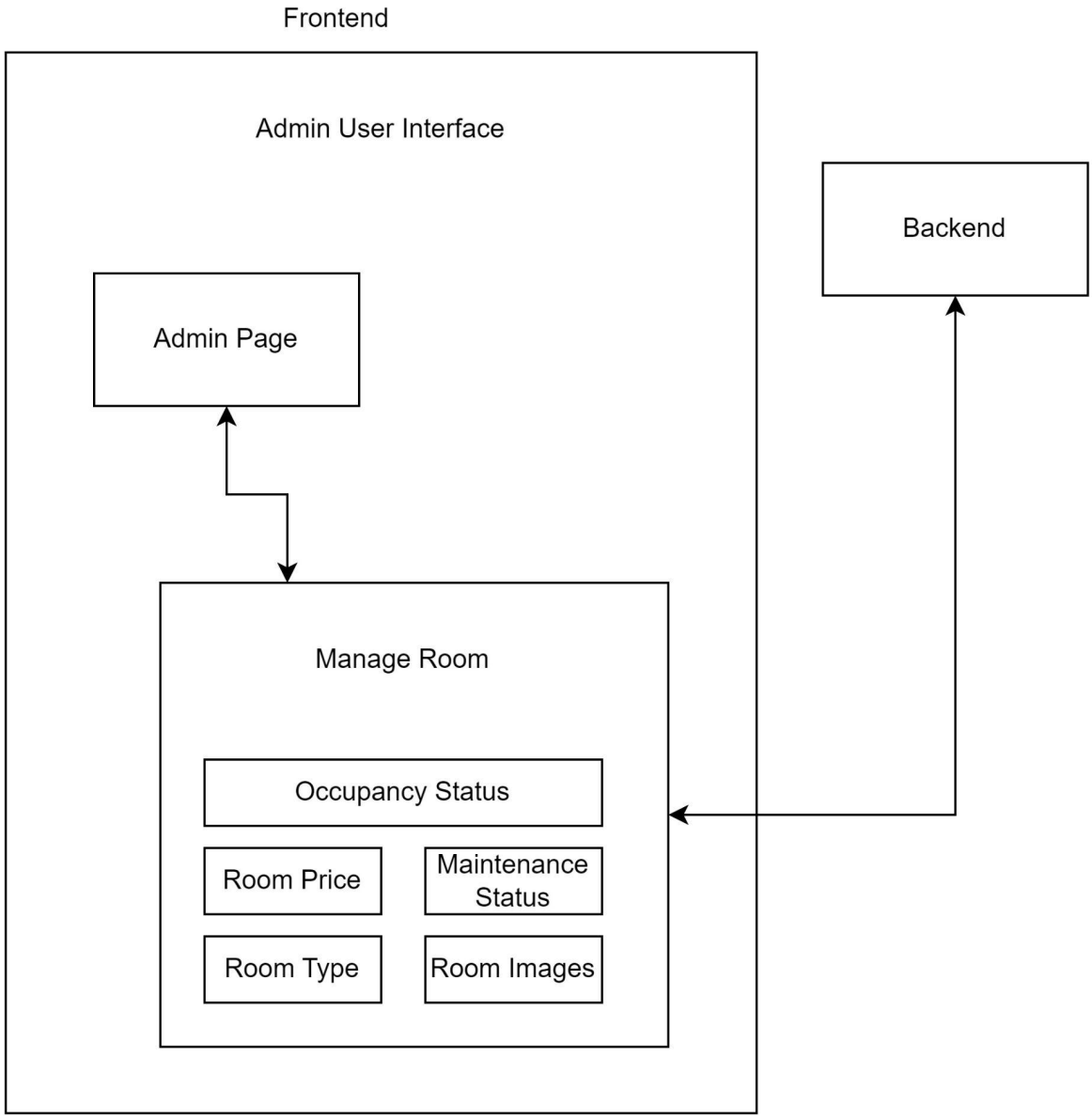


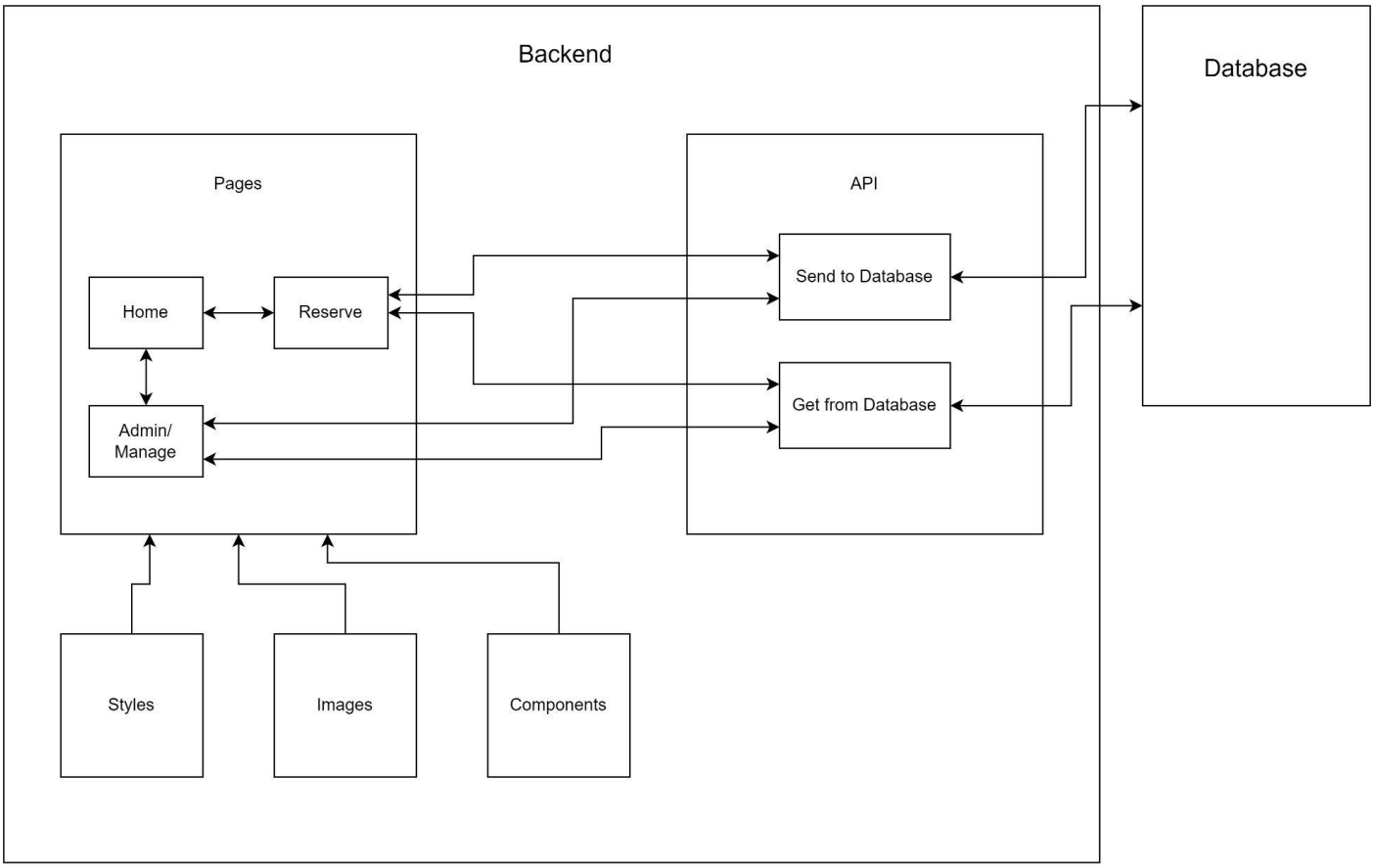
Fig 2. Admin View



2.3 Back End

NextJS will be used as the backend framework for React on the frontend. NextJS provides both client-side and server-side rendering for increased flexibility and speed, and adds the ability to implement API calls which will be used to communicate with the Firestore database. NextJS also provides increased image optimization, which will be important for the reservation service as each room listing will include several pictures.

Fig 3. Backend Architecture



2.4 Database

Firestore will be used as the database for the project. It is a flexible and scalable noSQL document database provided by Google Cloud Platform. It allows ease of data storage into collections and documents, which enables schema-less data modeling. It also offers a real time data synchronization feature, this makes it suitable real time info for clients using the web page.

3. Software Modules

3.1 Front End

3.1.1 User Interface Components:

React allows us to create reusable UI components for different parts of the hotel web app like header, footer, navigation, room listing, and reservation form.

3.1.2 State Management:

React provides a mechanism for managing and updating the app as well as handling dynamic data like user inputs, reservations, and availability which makes it possible for us to respond to changes and update the UI accordingly.

3.1.4 Styling:

React allows us to use CSS or CSS-in-JS to style components which makes customization easy thus creating an appealing and consistent user interface.

3.2 Back End

3.2.1 Routing:

NextJS provides a simple way to route the user to specific pages. The root directory of the NextJS project contains a pages/ folder, and any file placed in it will be accessible as a route.

3.2.2 API Routes:

Next.js also makes API implementation easy. Any file in the pages/api/ folder will be treated as an API endpoint instead of a normal page, allowing the user to send and receive requests from their custom API. This will allow us to send and receive data from our Firsestore database.

3.3 Database

3.3.1 Data Modeling

Fire store enables schema-less modeling, this allows for dynamic and flexible storage of data. Data is organized into collections and documents, providing the flexibility to adapt to changing data and structure without requiring a predefined schema.

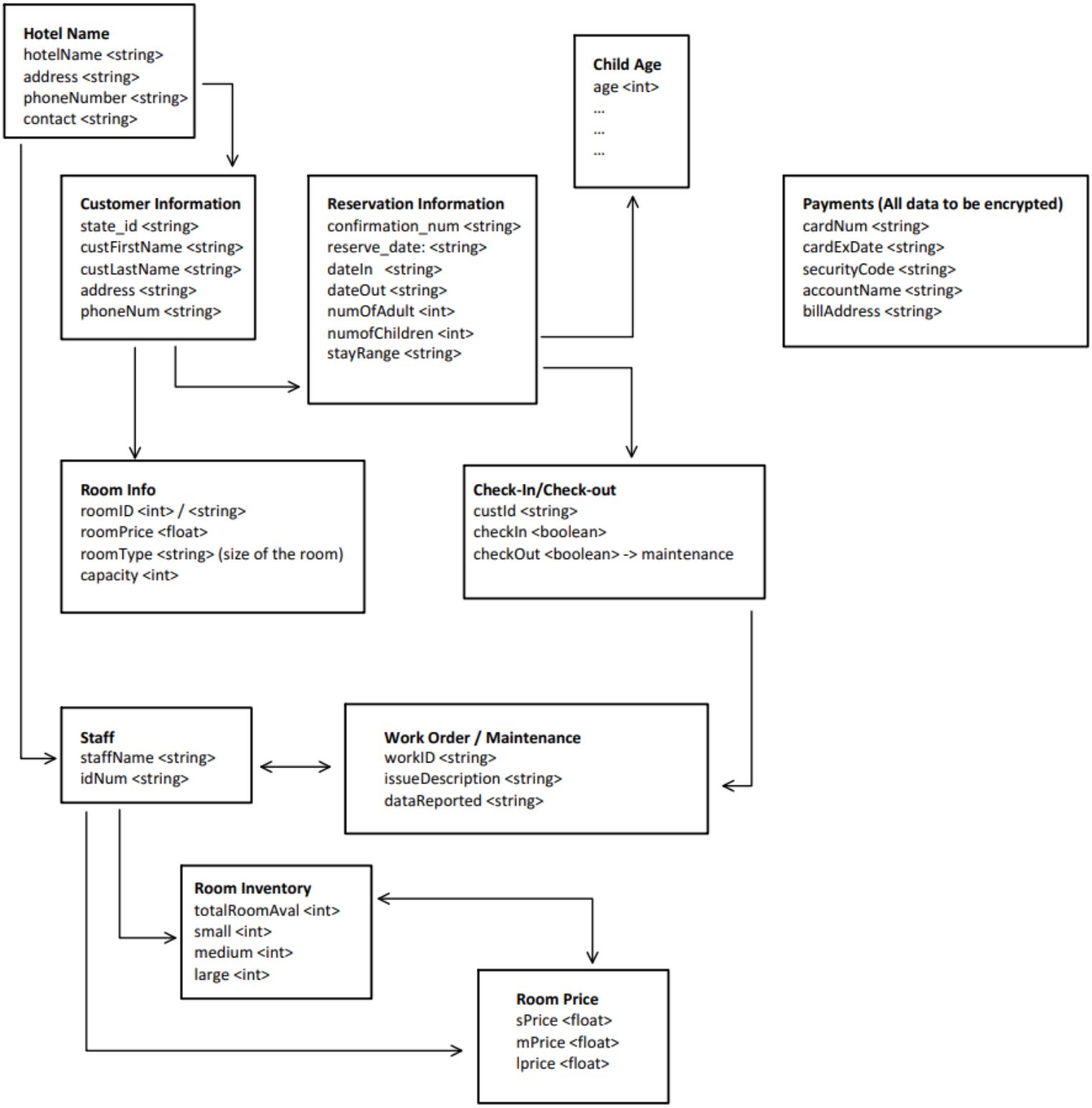
3.3.2 Data Storage and Retrieval

Firestore allows ease of data storage and retrieval. The intuitive API provided by Firestore allows creating, reading, updating, and deleting operations on documents and allows users to manipulate data easily.

3.3.2 Database Design

Since the database is designed using Firestore, it is designed in a way where there are multiple collections within each unique sub data collection. See diagram below for more details.

Fig 4. Hotel Database Design



4. Dynamic Models of Operational Scenarios (Use Cases)

4.1 Client Reservation

Allows clients to make reservations at the hotel. It involves the client selecting the desired check-in and check-out dates, choosing the room, providing personal details, and making the payment. The system verifies availability, calculates the price, and generates reservation confirmation.

4.2 Hotel Management

This covers the functionalities required for managing the hotel operations. It includes features like handling guests requests, monitoring room occupancy, and ensuring smooth operations throughout the hotel.

4.3 Addition to Rooms

Enables hotel management to add new rooms to the system. It involves specifying room details as room type, amenities, and assigning a unique identifier. The system updates the room inventory and ensures proper integration with other modules.

4.4 Price Management

Allows hotel managements to set and modify room prices based on demand and special offers. The system provides an interface to define pricing rules, updates rates, and ensures consistency across different channels and reservation platforms.

4.5 Check-in / Check-out

It covers the process of guest check-in and check-out as well as capturing guest information, verifying reservations, assigning rooms, and managing payments. The system updates the room inventory and handles any additional services or charges.

4.6 Room Inventory

Involves maintaining an up-to-date record of room availability and status. It tracks occupied rooms and ensures accurate information about the availability of rooms for reservations.

5. Acknowledgements

This documentation was referenced from Juan’s provided template. Some other structure formats was referenced from previous team Fetch.

6. Other References