**Detailed Design Document**

**Project Title: Hotel Management Availability System**

**1. Introduction**

The Hotel Management Availability System aims to provide a RESTful API for managing hotel product categories and checking availability for specific dates. The API will be built using ASP.NET Core and MongoDB as the database to store product and availability information.

**2. System Architecture**

The system architecture will consist of the following components:

* **ASP.NET Core API**: Responsible for handling HTTP requests and responses, routing, and invoking business logic.
* **MongoDB Database**: Stores hotel product categories and availability information.
* **Business Logic Layer**: Contains service classes responsible for interacting with the database and implementing business rules.
* **Unit Tests**: Ensures the correctness of API endpoints and business logic.

**3. API Endpoints**

The API will expose the following endpoints:

* **GET /api/products**: Retrieve a list of all available product categories.
* **GET /api/products/{id}**: Retrieve a specific product category by its ID.
* **POST /api/products**: Create a new product category.
* **PUT /api/products/{id}**: Update an existing product category by its ID.
* **DELETE /api/products/{id}**: Delete a product category by its ID.

**POST /api/availability**: Check availability for a specific product category and date range.

* + - **Sequence Diagram**

Client ProductsController ProductRepository MongoDB

| | | |

| GET /api/products (Request) | |

|------------------>| | |

| | GetAllProducts() | |

| |--------------------------> | |

| | | Query MongoDB |

| | |-------------------> |

| | | Retrieve Data |

| | |<------------------- |

| | | |

| | Products List (Response) | |

|<------------------| | |

* + - **Class Diagram:**

**+------------------+ +------------------+**

**| BaseProduct | | IProduct |**

**+------------------+ +------------------+**

**| - Id: string | | - CategoryName: string**

**| - CreatedAt: DateTime | - Capacity: int**

**+------------------+ | - PricePerNight: decimal**

**| |**

**| |**

**V V**

**+------------------+ +------------------+**

**| Product | |ProductRepository |**

**+------------------+ +------------------+**

**| - CategoryName: string | - \_products: IMongoCollection<Product>**

**| - Capacity: int | + GetAllProducts(): Task<IEnumerable<Product>>**

**| - PricePerNight: decimal | + GetProductById(string id): Task<Product>**

**+------------------+ | + CreateProduct(Product product): Task<Product>**

**| + UpdateProduct(string id, Product product): Task**

**| + DeleteProduct(string id): Task**

**+------------------+**

**4. Database Design**

The MongoDB database will consist of two collections:

* **Products Collection**: Stores information about hotel product categories, including ID, category name, capacity, and price per night.
* **Availabilities Collection**: Stores availability information for each product category, including the product ID, date, and availability status.

**5. Business Logic Layer**

The Business Logic Layer will include the following service classes:

* **ProductService**: Implements methods to interact with the Products collection, such as retrieving, creating, updating, and deleting product categories.
* **AvailabilityService**: Implements methods to check availability for specific product categories and date ranges.

**6. Unit Tests**

Unit tests will be written using NUnit framework to ensure the functionality of API endpoints and service classes. Tests will cover various scenarios, including success cases, edge cases, and error handling.

**7. Error Handling**

Error handling will be implemented at both API controller level and service level. The API will return appropriate HTTP status codes and error messages in case of invalid requests or server errors.

**8. Documentation**

API documentation will be provided using Swagger UI to describe the endpoints, request/response formats, and example usage.

A screenshot of a computer

Description automatically generated

**9. Deployment**

The API will be deployed to a cloud platform such as Azure or AWS, using Docker containers for easy deployment and scaling.