B. System Architecture

Group Name: TISM Tech
Ayaan Khan - i220832
Ayaan Mughal - i220861
Mishal Ali - i221291

---

1. Identifying Subsystems

Our system follows a **microservices-based architecture** with clearly defined subsystems corresponding to major functionalities in a hotel booking platform. The subsystems are:

- **User Service**: Handles user authentication, registration, login, and loyalty features.
- **Booking Service**: Manages room bookings, payment simulation, and reservation tracking.
- **Hotel Service**: Manages hotel and room listings, search and filter functionality, pricing updates, and reviews.
- **Frontend**: A React-based client application interfacing with backend services through REST APIs.

Each backend service is organized into a **layered structure** consisting of:

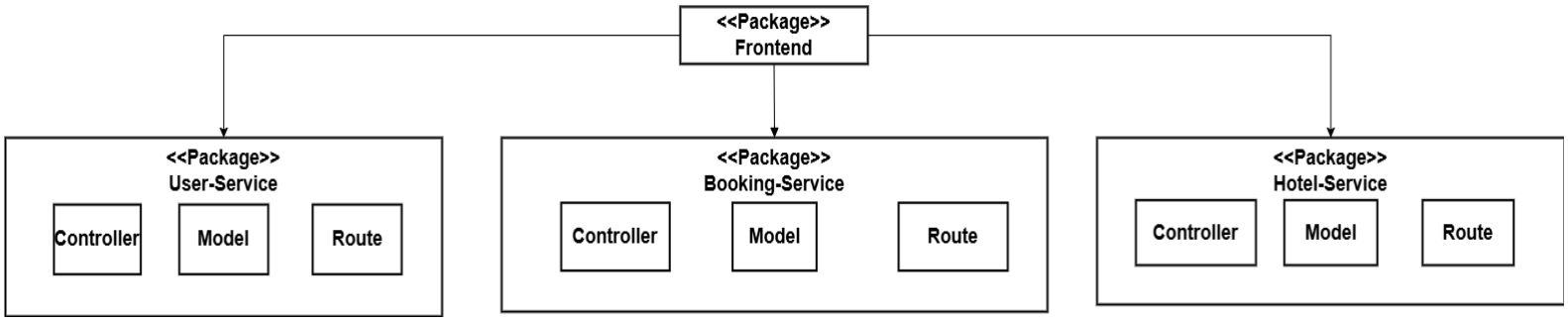
- Controller layer: Contains business logic.
- Route layer: Handles Routing requests.
- Model layer: Defines schema and interacts with MongoDB.

The frontend is modularized into:

- User Module
- Hotel Module
- Booking Module

These modules interact with their respective backend services via RESTful APIs.

UML Package Diagram of Subsystems:



---

2. Architecture Styles

The following architectural styles have been adopted in our system:

1. Microservices Architecture

Each core functionality (User, Hotel, Booking) is implemented as a **separate microservice**, ensuring modularity, scalability, and ease of deployment. Each service owns its data and operates independently.

2. Layered (3-tier) Architecture within Services

Each service follows a **layered structure**:

- **Controller**: Handles logic and validation
- **Route Layer**: Manages request routing
- **Model Layer**: Defines and interacts with MongoDB collections

---

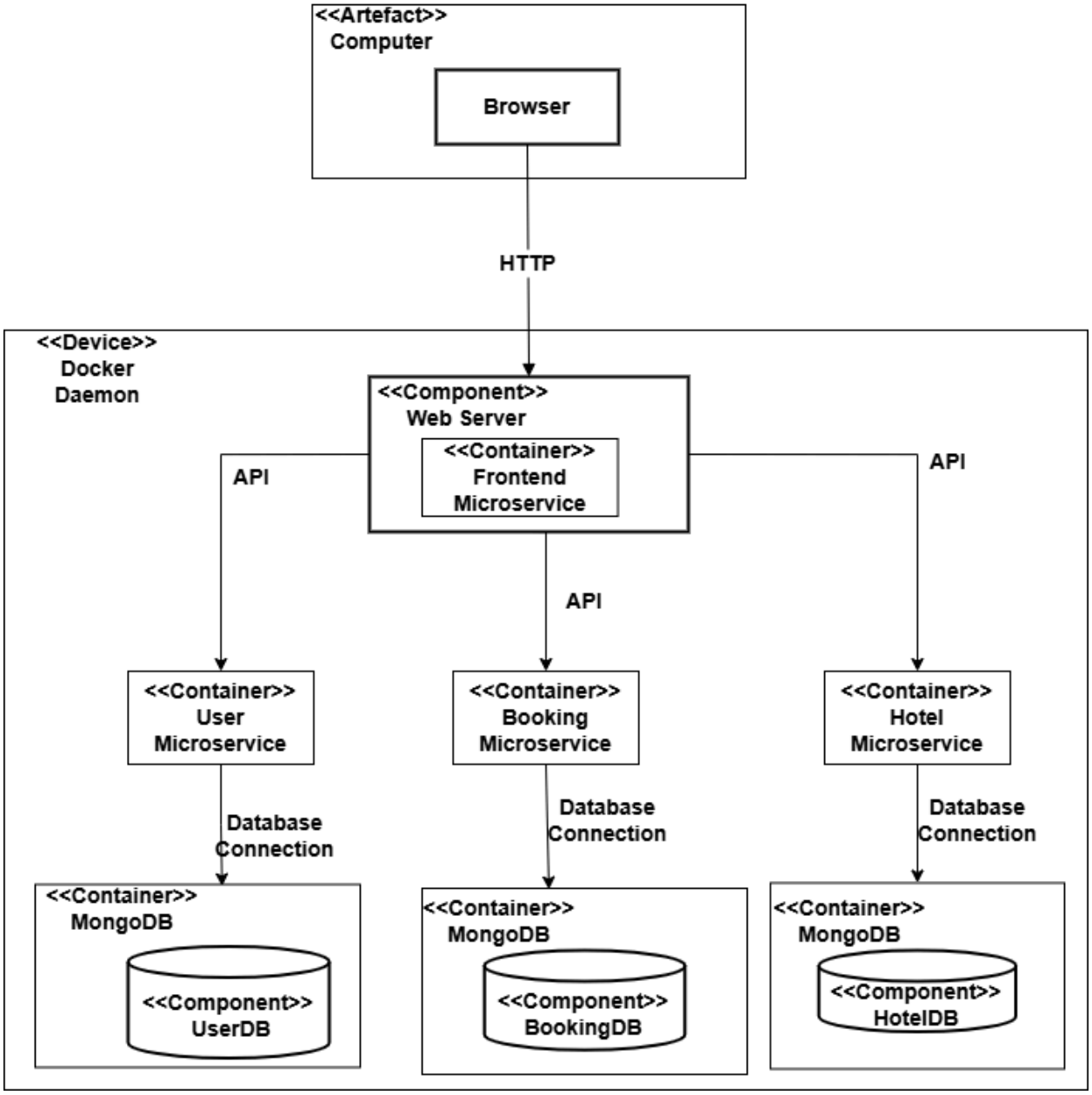
3. Deployment Diagram for Client Deployments

The system is fully containerized using **Docker**, and orchestrated using **Docker Compose** for local deployments. Each microservice and MongoDB instance runs in its own container. The frontend runs in a separate container served via a web server (Node).

Deployment Stack Overview:

- **Client Hardware**: Web Browser
- **Web Server Container**: Hosts and serves the React frontend
- **Docker Compose Cluster**: Runs backend microservices in isolated containers
- **MongoDB Instances**: Each service has a dedicated MongoDB database container

Deployment Diagram with Containers and Databases:



---

4. Component Diagrams

The component diagram illustrates how different parts of the system **logically interact** and which services are enhanced in this version:

Enhancement Highlights:

- **Booking Service** interfaces with:
    - **Hotel Service** for room data
    - **User Service** for authenticated booking
    - **Payment Gateway (Simulated)** for transaction flow
- **Hotel Service** includes a new **Pricing Engine** and **Filter/Search Module**
- **Frontend** interacts with backend services through:
    - User Module → User Service
    - Hotel Module → Hotel Service
    - Booking Module → Booking Service

Each service is also internally divided into layered components (Controller → Service → Model), and each is connected to its dedicated MongoDB instance.

Component Diagram with Frontend, Backend, and MongoDB: