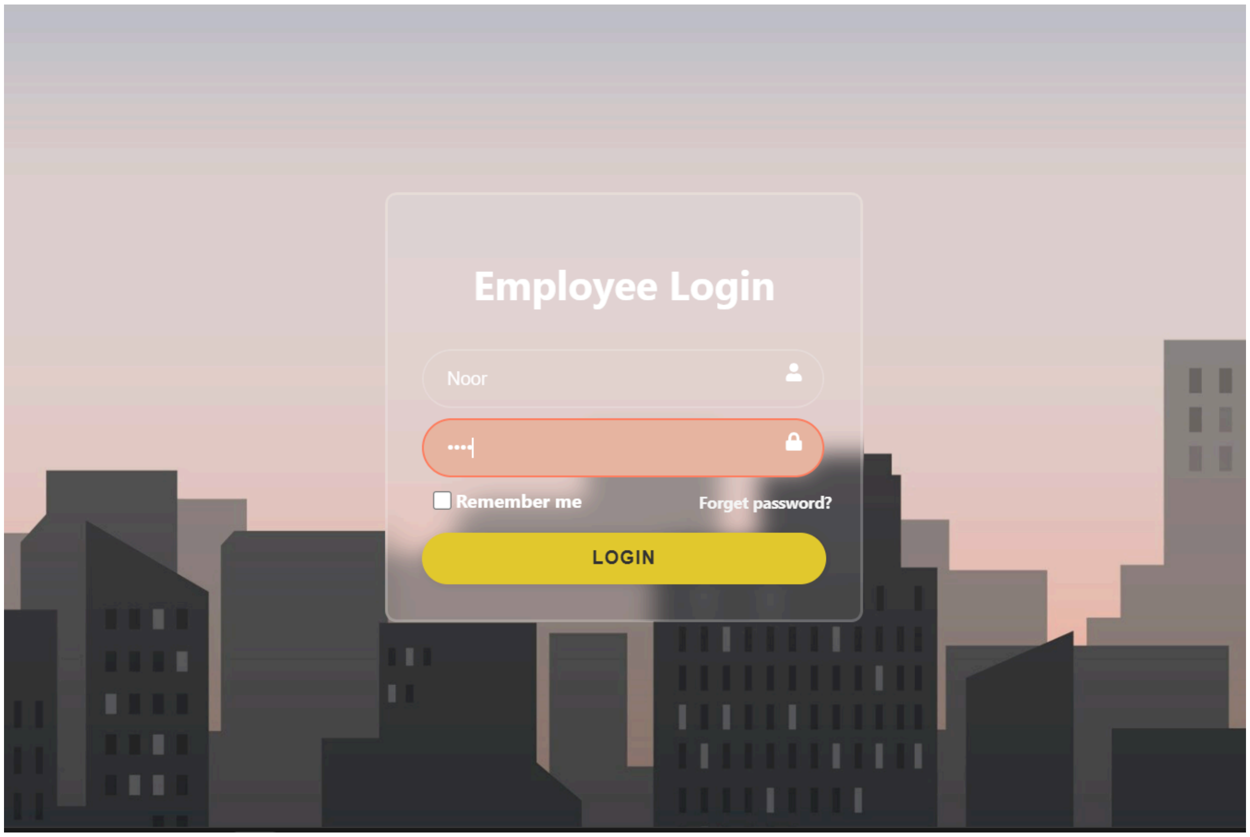


PayTrack

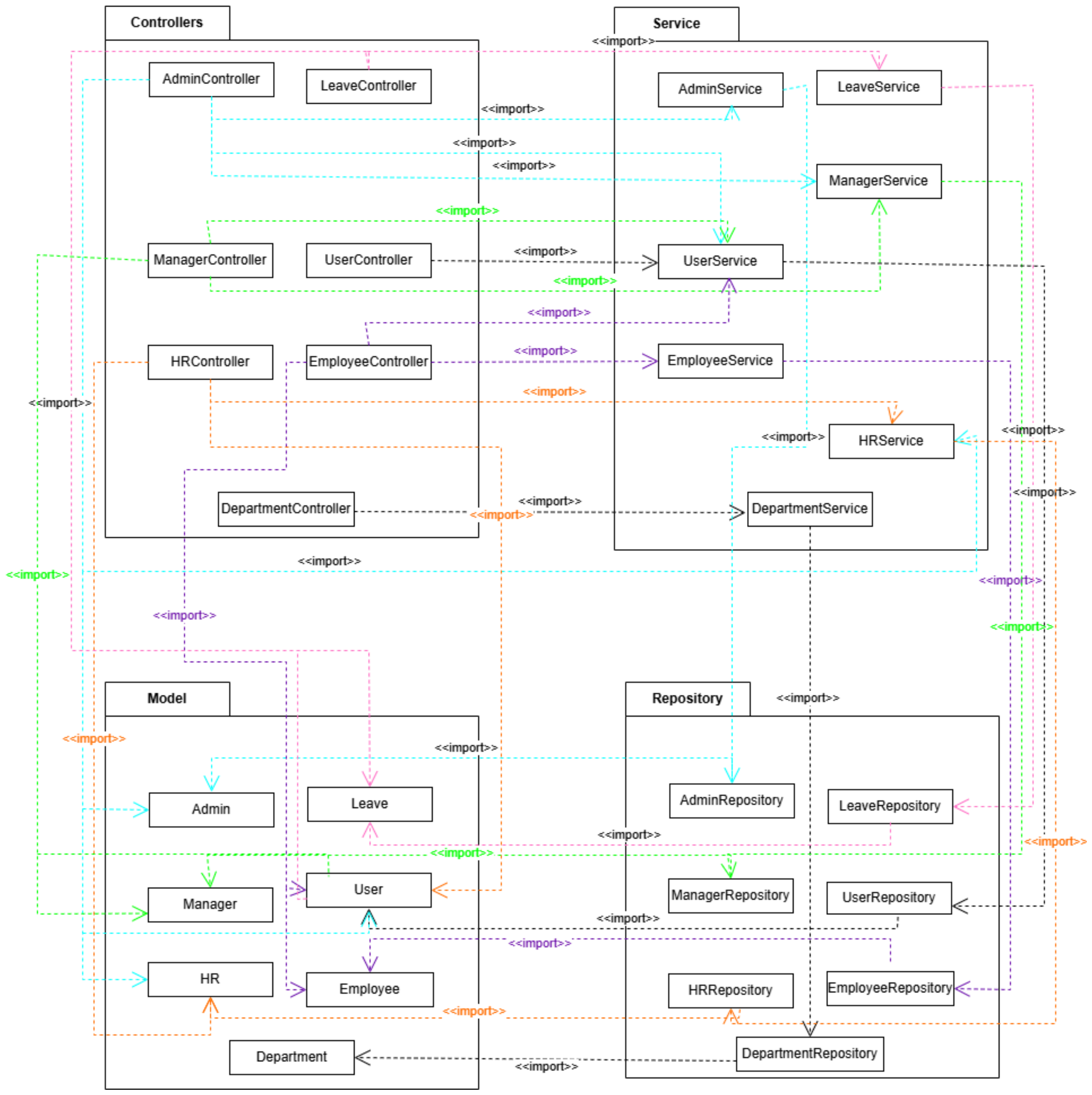
Rio

Noor Fatima – 22i-1036

Sara Akbar – 22i-0846

Abdul Munim – 22i-1080

Package Diagram



Controllers

- AdminController, LeaveController, ManagerController, UserController, HRController, EmployeeController, DepartmentController: These classes handle HTTP requests for specific entities or functionalities in the system. Each controller corresponds to a different part of the business logic (e.g., AdminController for administrative tasks, LeaveController for leave-related functionality).
- They import and interact with the service classes to perform necessary operations.

Services

- **AdminService, LeaveService, ManagerService, UserService, EmployeeService, HRService, DepartmentService**: These classes contain the core business logic for each entity or operation. They are directly called by their corresponding controllers.
- Services act as a bridge between the controller layer (handling HTTP requests) and the repository layer (accessing the database).

Model

- **Admin, Leave, Manager, User, HR, Employee, Department**: These are your data models (entities). They define the structure of the data that will be persisted in the database, with each model representing a specific entity within the system.
- These models are used by the services and repositories to process and store data.

Repository

- AdminRepository, LeaveRepository, ManagerRepository, UserRepository, HRRepository, EmployeeRepository, DepartmentRepository: These classes handle the data access logic. They interact with the database to perform CRUD (Create, Read, Update, Delete) operations.
- Repositories are used by services to fetch or store data, ensuring that the actual data persistence logic is separated from the business logic.

Connections Between Components

- **Controller to Service**: Controllers communicate with services to handle the business logic. For example, AdminController uses AdminService to manage administrative tasks.
- **Service to Repository**: Services interact with repositories to manage data. For instance, AdminService uses AdminRepository to interact with the database for administrative operations.

- **Model to Repository**: The repository classes typically interact with the models to fetch or persist data. The models represent the data structure, and repositories manage the actual database operations for these models.

Architecture Style

The architectural styles used in the PayTrack Employee Payroll & Leave Management System:

Role-Based Architecture

The system implements a role-based architectural style with distinct components for different user roles:

- Admin components (AdminLayout.js, AdminSidebar.js)
- HR components (HRLayout.js, HRSidebar.js)
- Manager components (ManagerLayout.js, ManagerSidebar.js)
- Employee components (EmployeeLayout.js, EmployeeSidebar.js)
- Each role has its own layout and navigation structure
- This separation ensures proper access control and role-specific functionality

Component-Based Architecture

The project follows React's component-based architecture with:

- Reusable UI components (card.js, tabs.js, badge.js)
- Layout components (AdminLayout.js, HRLayout.js, etc.)
- Feature-specific components (LeaveRequestCard.js, LeaveRequestTab.js)
- Components are organized in a hierarchical structure
- Each component is self-contained with its own styles (e.g., HRSidebar.css, EmployeeSidebar.css)

Single Page Application (SPA) Architecture

- Implemented using React Router (evident in App.js)
- Client-side routing for seamless navigation
- Centralized route management in Routes.js

Layered Architecture

The application is organized in distinct layers:

- Presentation Layer (components/)
- Routing Layer (routes/)
- Layout Layer (various Layout components)
- Business Logic Layer (feature-specific components)

Modular Architecture

- Clear separation of concerns with dedicated directories:
- components/ for UI elements
- routes/ for navigation
- pages/ for full page views
- Each module is independent and maintainable
- CSS modules for component-specific styling

Layout-Based Architecture

- Consistent layout structure across different user roles
- Each role has:
- A layout component (e.g., AdminLayout.js)
- A sidebar component (e.g., AdminSidebar.js)
- Role-specific styling (e.g., Admin.css, Manager.css)

Event-Driven Architecture

- React's state management and event handling
- Component interactions through props and events
- User interactions handled through event handlers

Service-Oriented Architecture (SOA)

- Backend services organized as independent services

- Frontend components consume these services
- Clear separation between frontend and backend

The combination of these architectural styles provides:

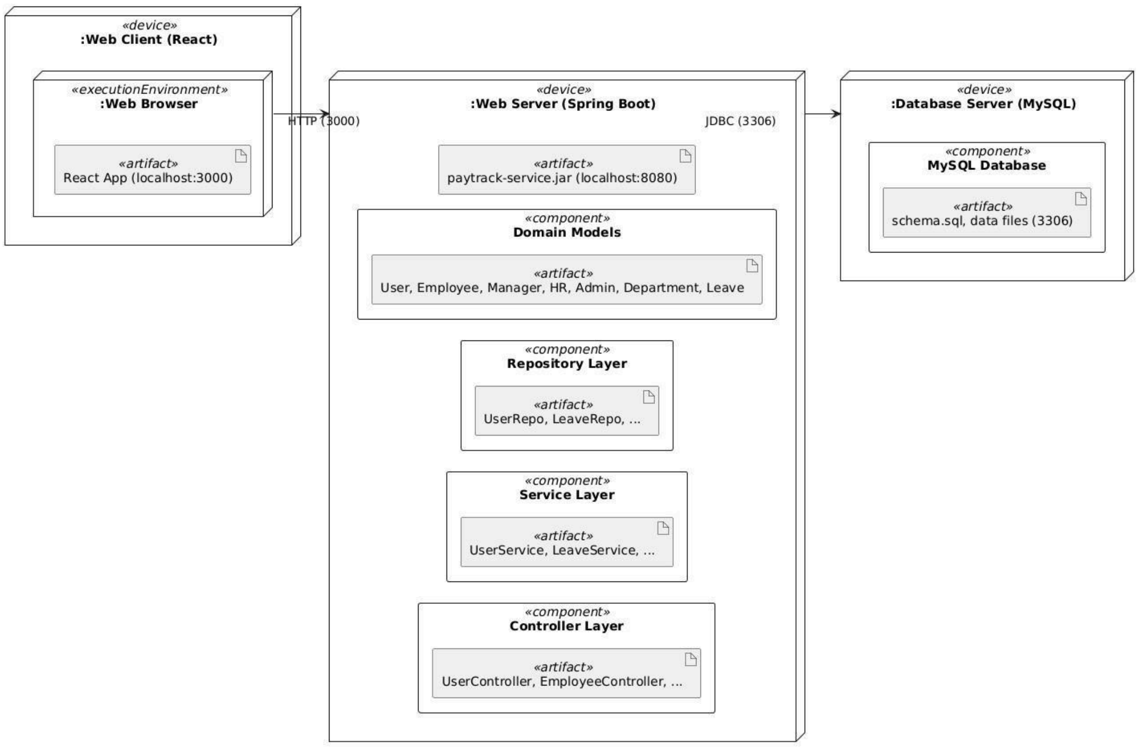
- Clear separation of concerns
- Role-based access control
- Maintainable and scalable codebase
- Reusable components
- Consistent user experience
- Efficient state management
- Clean code organization

Conclusion :

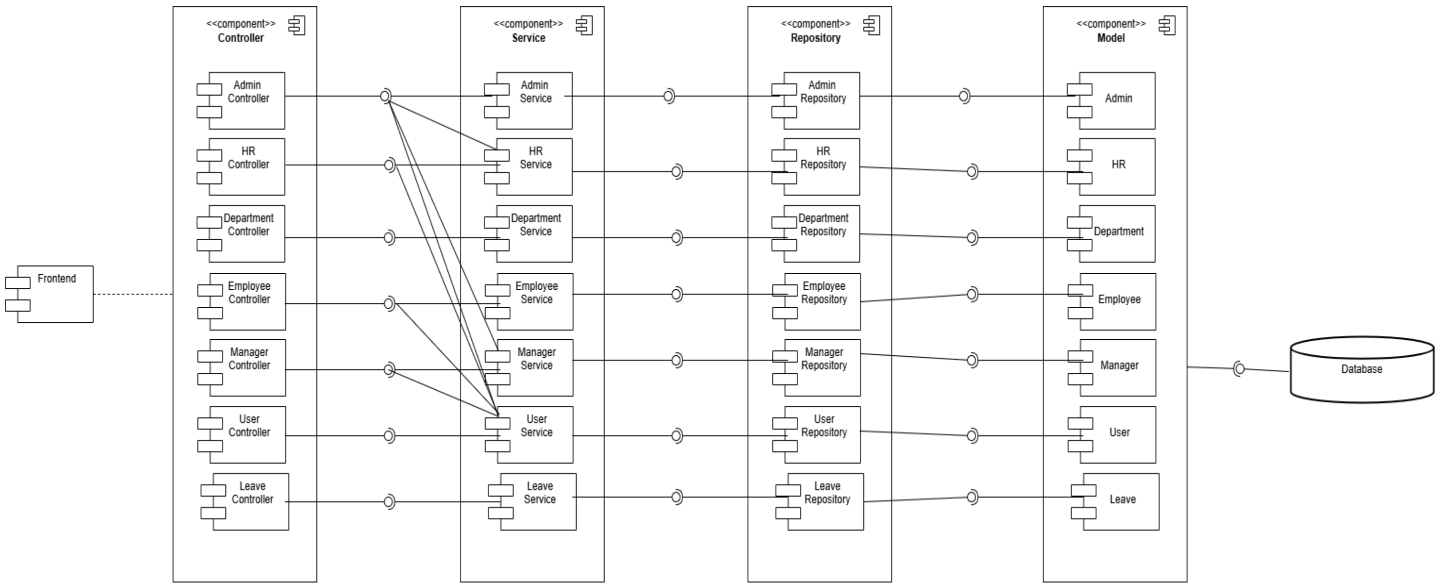
This architecture is well-suited for a payroll and leave management system as it:

- Supports multiple user roles
- Provides clear navigation
- Maintains security boundaries
- Allows for easy feature additions
- Ensures consistent user experience
- Facilitates maintenance and updates

Deployment Diagram



Component Diagram


<<component>>
Controller
Admin Controller
HR Controller
Department Controller
Employee Controller
Manager Controller
User Controller
Leave Controller
<<component>>
Service
Admin Service
HR Service
Department Service
Employee Service
Manager Service
User Service
Leave Service
<<component>>
Repository
Admin Repository
HR Repository
Department Repository
Employee Repository
Manager Repository
User Repository
Leave Repository
<<component>>
Model
Admin
HR
Department
Employee
Manager
User
Leave
Frontend
Database