# Employee Management System

## 1. Overview

The Employee Management System is a web-based application designed to manage employee data efficiently. This system supports various CRUD operations (Create, Read, Update, Delete) with role-based access control. The application leverages a microservices architecture to ensure scalability and robustness. Key functionalities include adding, viewing, editing, and deleting employee records, alongside features for sorting and pagination.

## 2. Technologies Used

- **Frontend**: Angular (for a responsive and interactive user interface)  
- **Backend**: Spring Boot (Java-based REST API for managing business logic)  
- **Database**: H2 Database (lightweight and easy-to-configure in-memory database)  
- **Cloud**: AWS (deployment and cloud integration)  
- **Architecture**: Microservices (to separate concerns and ensure modularity)

## 3. Features and Functionalities

### 3.1 User Interface

**- Header:** Includes navigation links:  
 - Home  
 - Add Employee  
**- Search Bar:** Allows searching for employee details by ID, name, or email.  
**- Sorting:** Supports sorting employee records by ID in ascending or descending order.  
**- Pagination:** Manages the number of employee records displayed per page.

### 3.2 Employee Operations

- **Add Employee**: Accessible via the "Add Employee" header. Users can add new employee details, including their name and email.  
- **View Employee Details**: Displays the details of individual employees.  
- **Edit Employee Details:** Enables modification of existing employee records.  
- **Delete Employee Records:** Allows removal of employee details from the system.

### 3.3 Role-Based Access Control

- **Admin Access:** - Can perform all operations, including adding, viewing, editing, and deleting employee details.  
**- User Access:**  
 - Limited to viewing employee details. Editing and deleting are restricted.

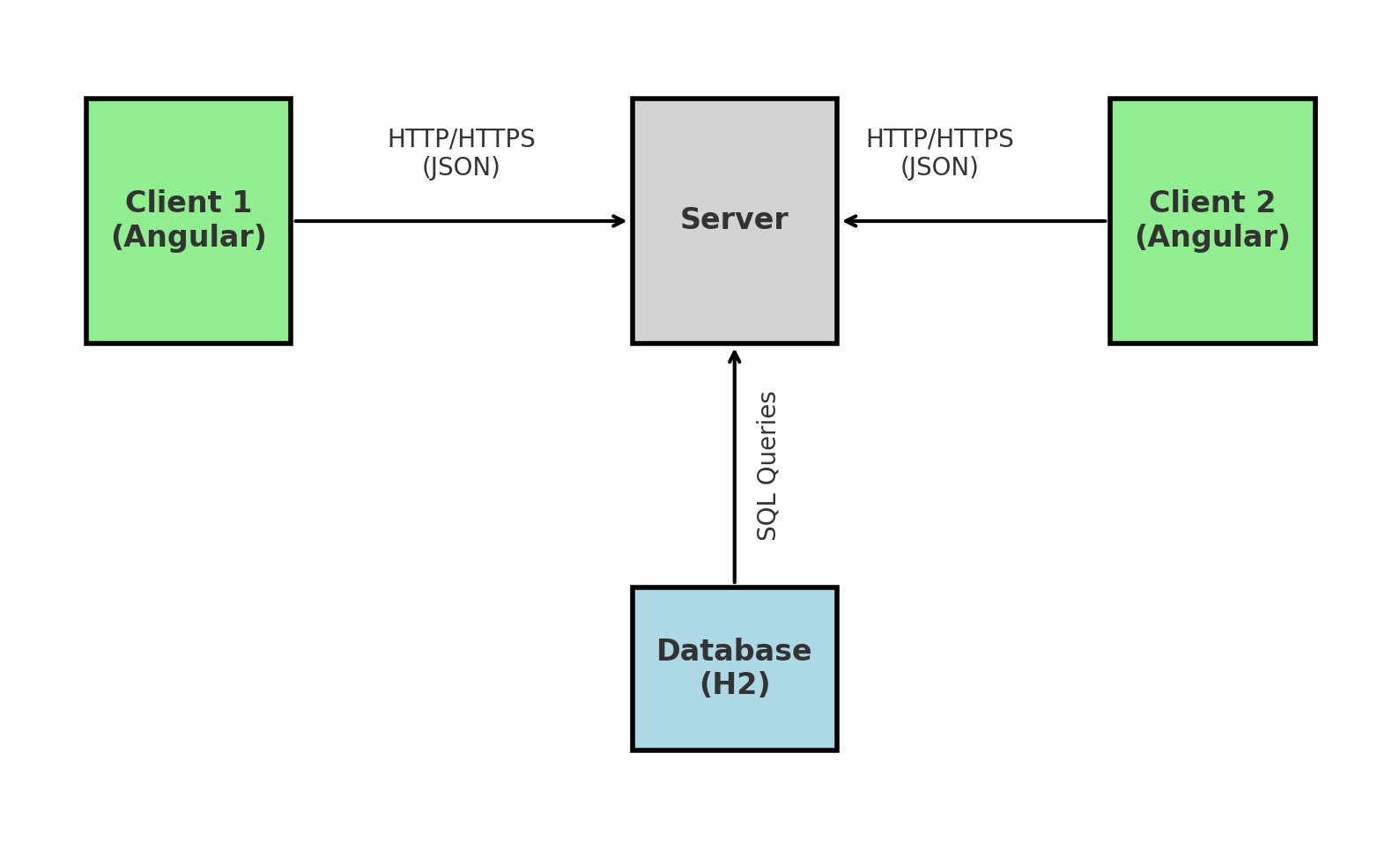
## 4. Architecture and Design

### 4.1 System Architecture

- **Frontend**: Angular application consumes APIs from the backend.  
- **Backend**: Spring Boot provides REST APIs for managing employee records.  
- **Database**: H2 database stores employee data. It is ideal for development and testing.  
- **Cloud Integration:** AWS handles deployment, scaling, and monitoring of the application.  
- **Microservices:** Separates authentication, employee management, and access control services.

### 4.2 Client-Server Architecture

The Employee Management System follows a client-server architecture to ensure separation of concerns and scalability:  
- **Client:**  
 - Built using Angular to provide an interactive and user-friendly interface.  
 - Sends HTTP requests to the server for performing CRUD operations.  
**- Server:**  
 - Developed using Spring Boot to handle business logic and process client requests.  
 - Provides RESTful APIs for interacting with the H2 database.  
- Communication:  
 - All communications between client and server occur over HTTP/HTTPS using JSON as the data exchange format.  
 - Secure APIs with role-based access control are implemented using JWT (JSON Web Tokens).



**Client-Server Architecture**

### 4.3 User Flows

**1. Sign-Up Page:**  
 - New users can register by providing their credentials.  
**2. Login Page:**  
 - Registered users can log in to access the system based on their roles.  
**3. Admin Operations Flow:**  
 - Admins can add, view, edit, and delete employee records.  
**4. User Operations Flow:**  
 - Users can log in and view employee details but cannot modify them.

## 5. Deployment Details

- Hosting Platform: AWS  
- Containerization: Dockerized microservices for isolated deployment.  
- Continuous Integration/Continuous Deployment (CI/CD): Configured pipelines for automated testing and deployment.

## 6. Security and Authorization

- **Authentication:** Ensures secure login for admin and users.  
- **Role-Based Access Control:** Differentiates operations based on user roles.  
- **API Security:** Secured APIs using JWT (JSON Web Tokens).

## 7. Development and Testing Workflow

- Agile Methodology: Regular sprints to develop and deliver features incrementally.  
- Unit Testing: Comprehensive tests for both frontend (Angular) and backend (Spring Boot).  
- Integration Testing: Ensures seamless communication between microservices.  
- UI Testing: Verifies functionality of navigation, form validation, and role-specific actions.

## Conclusion and Future Enhancements

The Employee Management System is a robust solution for managing employee data with role-based access. Its modular microservices architecture ensures scalability, while the use of AWS enhances deployment and availability.

### Future Enhancements

- Integrate a larger database like MySQL or PostgreSQL for production use.  
- Implement analytics dashboards for admins.  
- Add email notifications for employee updates.  
- Extend support for multi-language UI.