**explanation of each and every step in Employee-management-service**

=======================================================================

1. Project Setup

Tool: Spring Initializr (https://start.spring.io/)

Dependencies:

Spring Web

Spring Data JPA

Thymeleaf (for UI, optional)

Spring Boot DevTools

Validation (Jakarta Bean Validation)

H2/MySQL/PostgreSQL (Database)

Spring Boot Actuator (Optional)

Springdoc OpenAPI/Swagger UI

📂 2. Project Structure

employee-management-service/

├── src/

│ └── main/

│ ├── java/

│ │ └── com/example/employeemanagement/

│ │ ├── controller/ # Web & REST Controllers

│ │ ├── exception/ # Global Exception Handling

│ │ ├── model/ # Employee Entity

│ │ ├── repository/ # JPA Repository

│ │ ├── service/ # Service Layer

│ │ └── EmployeeManagementApplication.java

│ └── resources/

│ ├── templates/ # Thymeleaf HTML files

│ ├── static/ # CSS, JS (optional)

│ └── application.properties

└── pom.xml

🧑‍💼 3. Entity: Employee

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotBlank

private String firstName;

@NotBlank

private String lastName;

@Email

private String email;

private String department;

// Getters and Setters

}

🧬 4. Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

⚙️ 5. Service Layer

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repository;

public List<Employee> getAllEmployees() { return repository.findAll(); }

public Employee saveEmployee(Employee emp) { return repository.save(emp); }

public void deleteEmployee(Long id) { repository.deleteById(id); }

public Optional<Employee> getById(Long id) { return repository.findById(id); }

}

🌐 6. Web Controller (Thymeleaf UI)

@Controller

public class WebController {

@Autowired

private EmployeeService service;

@GetMapping("/") public String viewHome(Model model) {

model.addAttribute("list", service.getAllEmployees());

return "list";

}

@GetMapping("/add") public String addForm(Model model) {

model.addAttribute("employee", new Employee());

return "add";

}

@PostMapping("/employees")

public String saveEmployee(@ModelAttribute Employee employee) {

service.saveEmployee(employee);

return "redirect:/";

}

@GetMapping("/delete/{id}")

public String delete(@PathVariable Long id) {

service.deleteEmployee(id);

return "redirect:/";

}

}

🔄 7. REST API Controller

@RestController

@RequestMapping("/api/v1/employees")

public class EmployeeRestController {

@Autowired

private EmployeeService service;

@GetMapping

public List<Employee> getAll() { return service.getAllEmployees(); }

@PostMapping

public ResponseEntity<?> save(@Valid @RequestBody Employee employee, BindingResult result) {

if (result.hasErrors()) {

List<String> errors = result.getFieldErrors().stream()

.map(e -> e.getField() + ": " + e.getDefaultMessage()).toList();

return ResponseEntity.badRequest().body(errors);

}

return ResponseEntity.ok(service.saveEmployee(employee));

}

@DeleteMapping("/{id}")

public void delete(@PathVariable Long id) {

service.deleteEmployee(id);

}

}

⚠️ 8. Global Exception Handling

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(MethodArgumentNotValidException.class)

public ResponseEntity<Map<String, String>> handleValidation(MethodArgumentNotValidException ex) {

Map<String, String> errors = new HashMap<>();

ex.getBindingResult().getFieldErrors()

.forEach(err -> errors.put(err.getField(), err.getDefaultMessage()));

return ResponseEntity.badRequest().body(errors);

}

}

🛠️ 9. Swagger Integration

Add dependency:

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.2.0</version>

</dependency>

Visit: http://localhost:8080/swagger-ui/index.html

🧪 10. Test API with Swagger or Postman

Sample POST request:

{

"firstName": "Radha",

"lastName": "Aluri",

"email": "radha@gmail.com",

"department": "IT"

}

💡 Swagger Note

Swagger is a documentation and testing tool.

It doesn't store data, but it calls REST endpoints (which store, update, or delete in the database).

==================================================================================================================

**📘 Employee Management Service – Documentation**

📌 Project Summary

The Employee Management Service is a full-stack Spring Boot application that allows users to:

✅ Add new employees

✅ View all employees

✅ Delete employees

✅ View employees via both web (Thymeleaf) and REST APIs

✅ Validate employee input fields

✅ Document APIs using Swagger (OpenAPI)

**🔧 Technologies Used**

Java 17+

Spring Boot

Spring Web (MVC + REST)

Spring Data JPA

Hibernate (JPA Provider)

MySQL / H2 (in-memory) database

Thymeleaf (UI templates)

Swagger UI (API Testing + Docs)

Jakarta Validation (Field validations)

**🧱 Features**

Feature Web UI (Thymeleaf) REST API Validations Swagger UI

List Employees ✅ ✅ N/A ✅

Add Employee ✅ ✅ ✅ ✅

Delete Employee ✅ ✅ N/A ✅

Email Validation ✅ ✅ ✅ ✅

Global Error Handling ❌ (Basic) ✅ ✅ ✅

📁 Project Structure Overview

com.example.employeemanagement

├── controller → Web and REST Controllers

├── exception → Global Exception Handling

├── model → Entity (Employee)

├── repository → EmployeeRepository (JPA)

├── service → Business logic

├── resources/templates → Thymeleaf HTML (list.html, add.html)

└── application.properties

📦 How to Run the Project

✅ 1. Prerequisites

Java 17+

Maven

IDE (IntelliJ / VS Code / Eclipse)

**▶️ 2. Steps to Run**

bash

git clone https://github.com/repo/employee-management-service.git

cd employee-management-service

bash

**# Build and run the app**

mvn spring-boot:run

🌐 3. Access App

Web UI: http://localhost:8080

Swagger UI: http://localhost:8080/swagger-ui/index.html

🧑‍💼 Employee Entity

java

@Entity

public class Employee {

@Id @GeneratedValue

private Long id;

@NotBlank(message = "First Name is required")

private String firstName;

@NotBlank(message = "Last Name is required")

private String lastName;

@Email(message = "Invalid email format")

private String email;

private String department;

}

🔌 REST Endpoints (via Swagger)

Method URL Description

GET /api/v1/employees Get all employees

POST /api/v1/employees Create new employee

DELETE /api/v1/employees/{id} Delete employee by ID

🖥️ Web UI Screens

GET / → Homepage with employee list

GET /add → Add Employee form

POST /employees → Submit new employee

GET /delete/{id} → Delete employee

⚠️ Validation & Error Handling

Validation: Uses @NotBlank, @Email in Employee model

Error Handling: Catches and returns JSON error messages using @ControllerAdvice

Example API Error Response:

{

"firstName": "First Name is required",

"lastName": "Last Name is required"

}

🧪 Testing

You can test APIs using:

✅ Swagger UI: http://localhost:8080/swagger-ui/index.html

✅ Postman

✅ curl

Example curl:

curl -X POST http://localhost:8080/api/v1/employees \

-H "Content-Type: application/json" \

-d '{"firstName":"Radha","lastName":"Aluri","email":"radha@gmail.com","department":"IT"}'

📌 Conclusion

This project helps you:

Understand how to build a full-stack CRUD app using Spring Boot

Understand how to use REST APIs and validation

Integrate Swagger for documentation and testing

======================================================================================================

**✅ How the Employee Management System Benefits Employees**:

1. 🎯 Centralized Information

All your personal info (name, email, department, etc.) is stored safely in one place.

If you update your email or change departments, it can be updated easily.

"No more Excel sheets or paper records — everything is organized."

2. ⏱️ Faster HR Processes

If you apply for leave, get promoted, or get transferred — your data is already there.

HR can quickly check your role, department, experience, etc.

3. 📋 Accurate Records

The system avoids human error.

Your data is validated (e.g., no invalid email, missing name, etc.)

"This prevents issues like wrong name on certificates or salary slips."

4. 🔐 Security & Privacy

Your data is stored securely using Spring Boot and a database like MySQL.

Only authorized HR or admins can access it.

5. 📊 Reports and Appraisals

Your attendance, department history, and performance can be tracked easily.

Makes appraisals and performance reviews more transparent.

6. 🌐 Possible Employee Portal Add-on

If expanded, this system can allow employees to:

View their profile

Update certain fields (like address)

Apply for leave

Download payslips

🧠 So, while the EMS does not "run programs", it:

Stores structured employee data

Helps automate and simplify company processes

Indirectly improves the employee experience

=======================================================

# Employee Management System

A full-stack web application built using \*\*Spring Boot\*\* to manage employee information such as first name, last name, email, and department.

## 📌 Project Features

- 📝 Add New Employee

- 🧾 View Employee Details

- ✏️ Edit Employee Info

- ❌ Delete Employee

- 🌐 REST API with Swagger Documentation

- ✅ Form validation with error handling

## 💻 Technologies Used

- Java 17+

- Spring Boot

- Spring MVC (Web Layer)

- Spring Data JPA

- Thymeleaf (HTML Template Engine)

- MySQL (Database)

- Hibernate (ORM)

- Swagger / OpenAPI (API Documentation)

**Configure MySQL in application.properties:**

properties

spring.datasource.url=jdbc:mysql://localhost:3306/employeedb

spring.datasource.username=root

spring.datasource.password=root

Build and run the project:

mvn spring-boot:run

Open in browser:

UI: http://localhost:8080

API Swagger Docs: http://localhost:8080/swagger-ui/index.html

✅ API Endpoints (Sample)

Method Endpoint Description

GET /api/v1/employees List all employees

POST /api/v1/employees Add a new employee

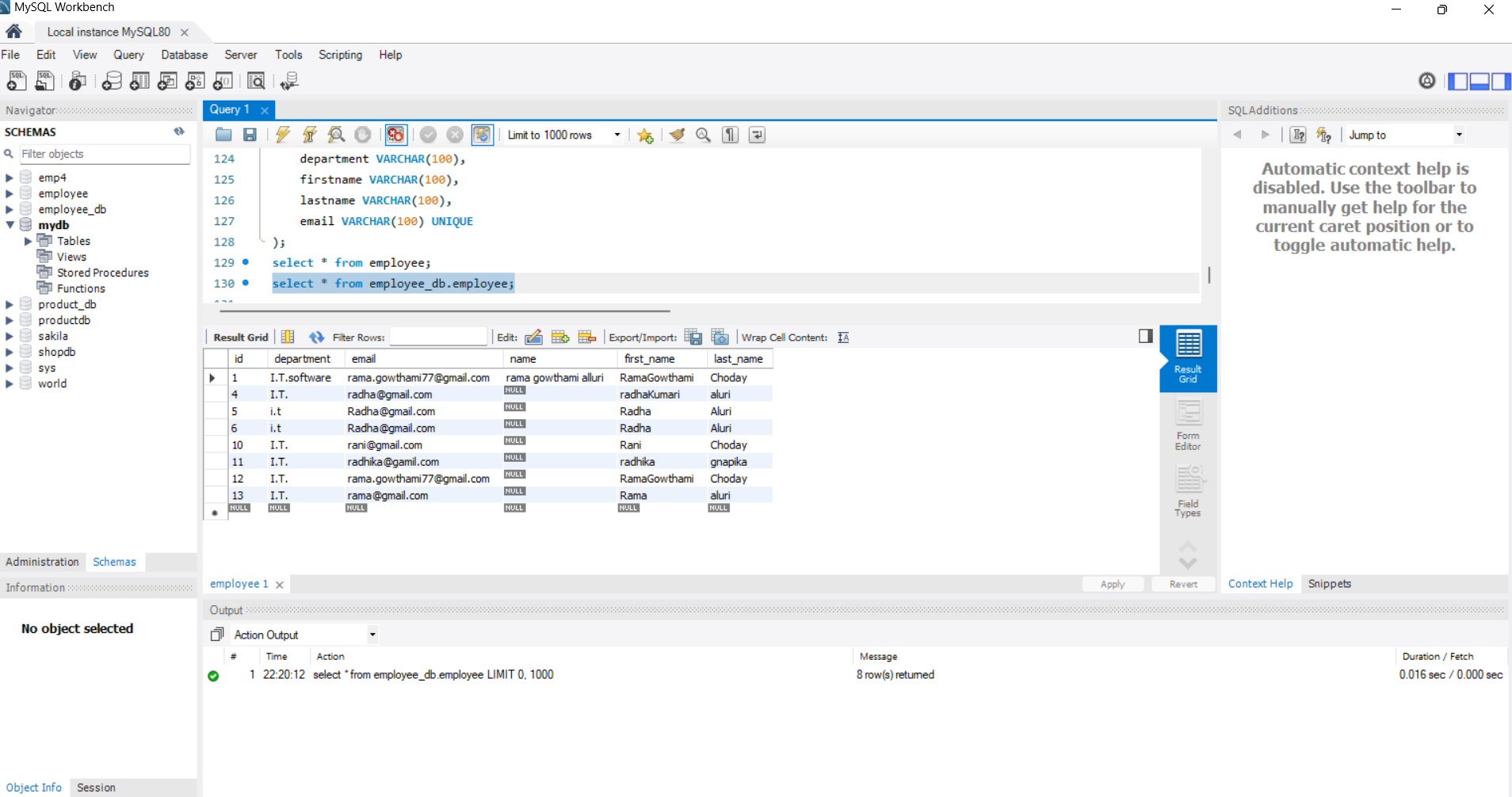
DELETE /api/v1/employees/{id} Delete an employee

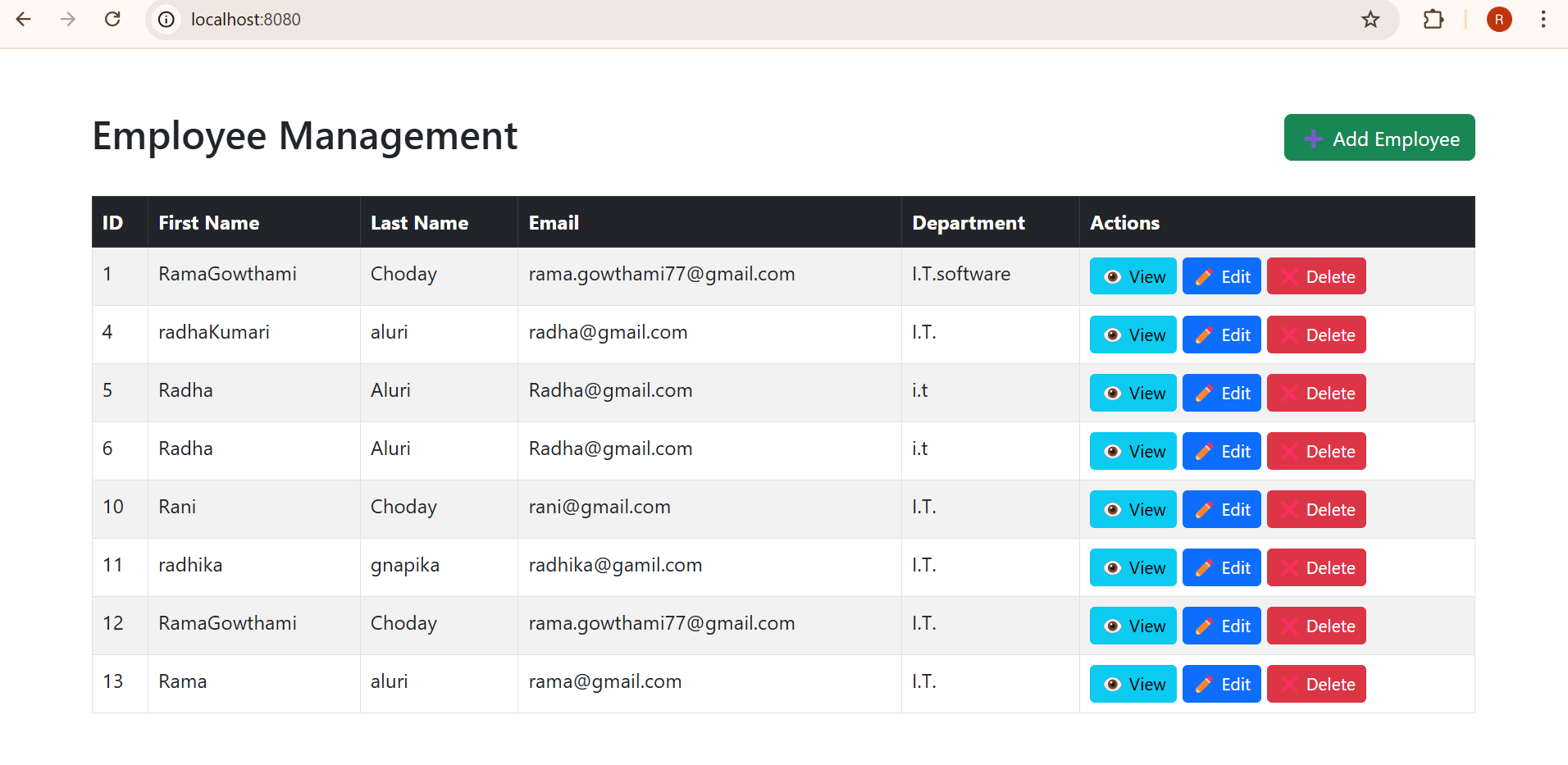
🛠️ Validations

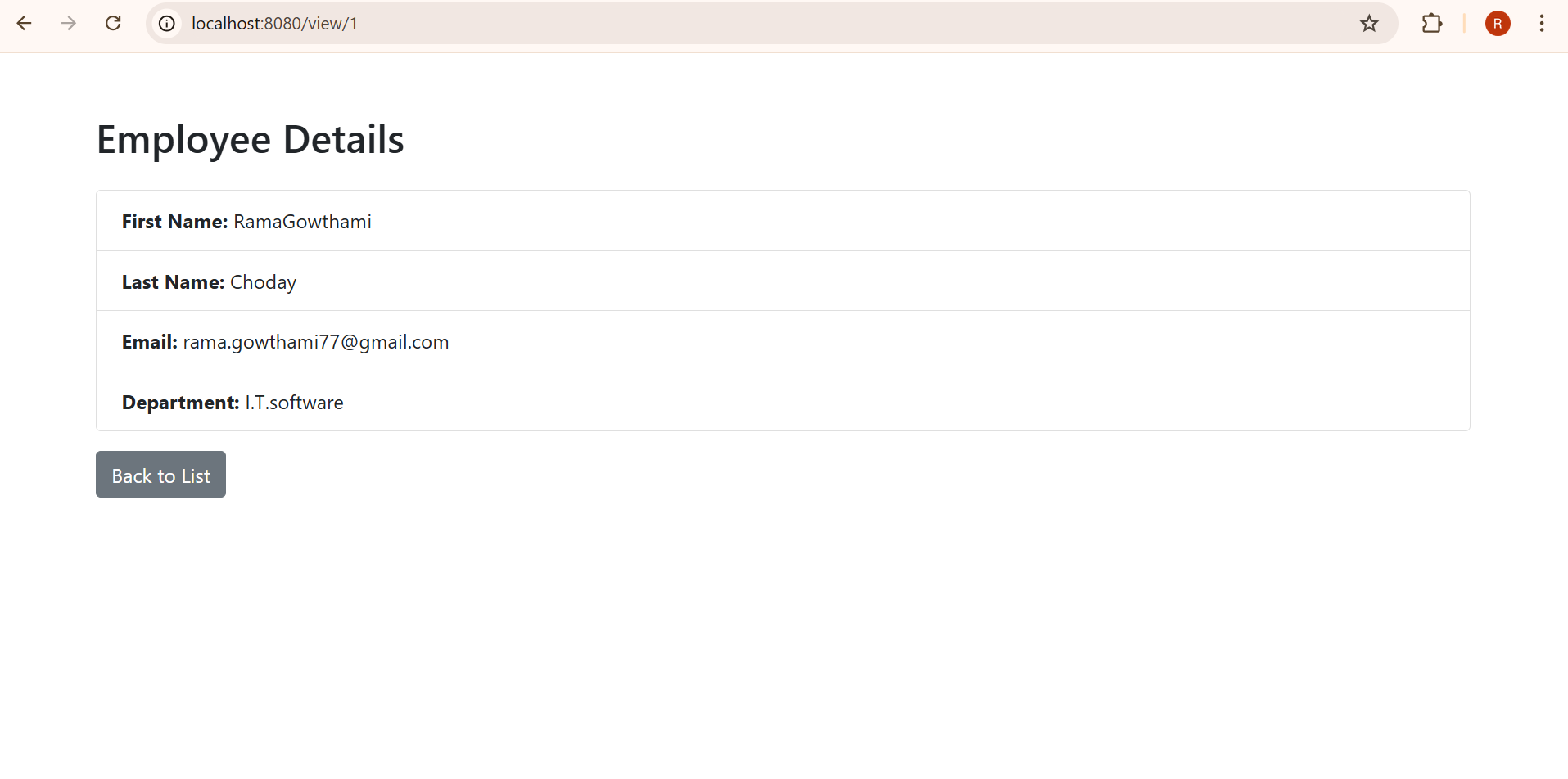
First Name: Required

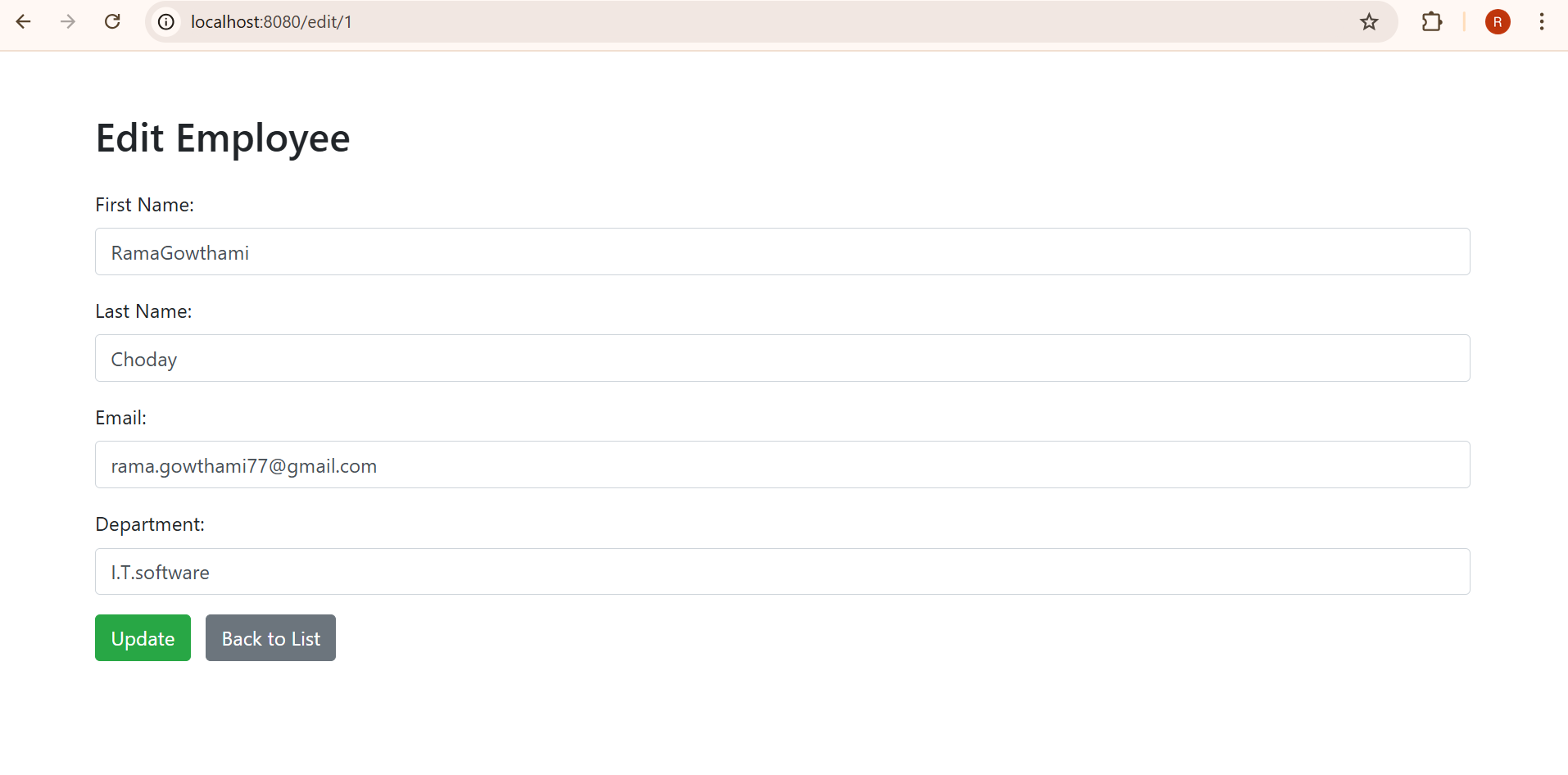
Last Name: Required

Email: Valid email format required

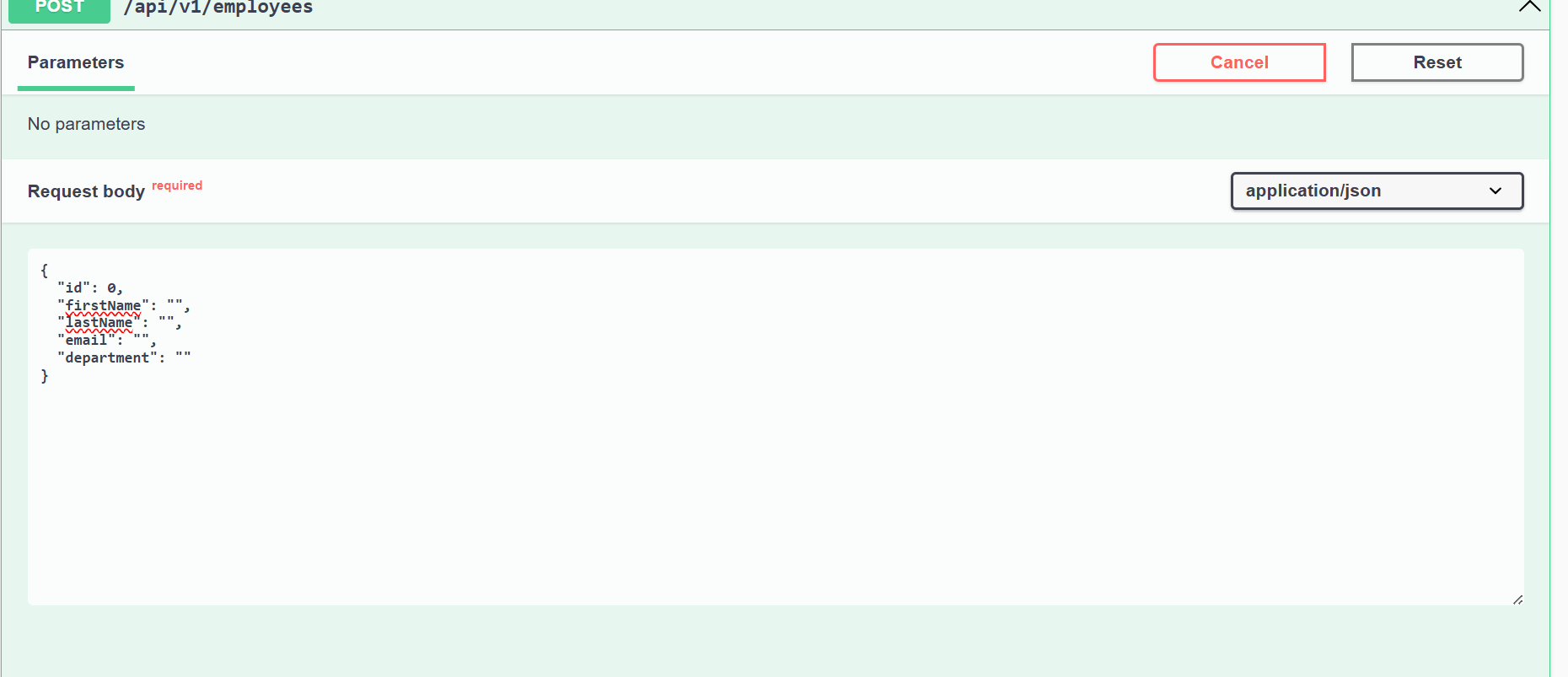


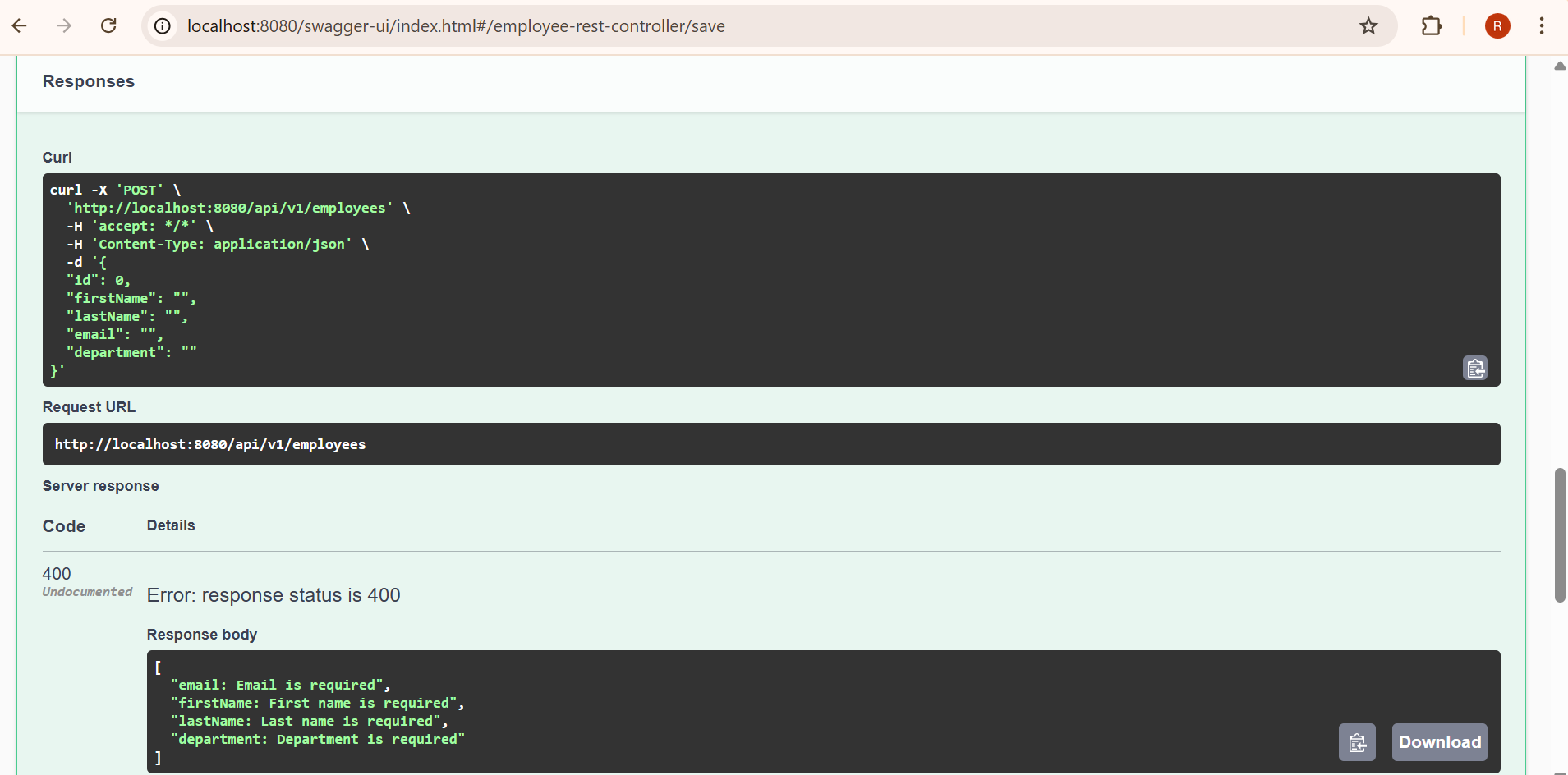












🙋‍♀️ Author

Made by [RamaGowthami Alluri].