**Food Delivery Project Summary**

This Food Delivery project is designed to create a robust and interactive web application for managing restaurants and dishes, featuring both an admin and end-user interface. The project utilizes a modern tech stack with Angular for the frontend and Spring Boot with microservices for the backend, ensuring scalable and responsive design, as well as secure data handling. Below is an overview of the project requirements and key tasks involved, along with relevant technologies and their usage in the industry.

**Project Overview:**

1. **Admin Frontend (Angular)**:
   * Build a web interface for admin users to add and manage restaurants and dishes.
   * Implement authentication for secure access, allowing only authorized admins to make changes.
2. **End-User Web Application (Angular)**:
   * Create a user-friendly interface to display restaurants and available dishes.
   * Allow users to browse and interact with the content seamlessly.
3. **Database Design and Management (SQL or MongoDB)**:
   * Design a well-structured database schema to store restaurant and dish data, using either MySQL or MongoDB.
   * Establish relationships between tables (primary and foreign keys) for efficient data management.
4. **Java Backend with Spring Boot (Microservices)**:
   * Develop backend microservices to handle business logic and data transactions.
   * Enable secure and efficient communication between frontend and backend using HTTP requests.
5. **Automation with Jenkins**:
   * Define a Jenkinsfile for continuous integration and delivery (CI/CD) of both Angular and Spring Boot projects.
   * Ensure automated builds for rapid testing and deployment.

**Key Tasks and Technologies:**

* **Task 1: Angular Components, Routing, Services, and AuthGuard**
  + **Usage**: Industry-standard techniques to create modular, single-page applications. AuthGuard ensures secure access control.
  + **Objective**: Enable admin authentication, create forms for adding restaurants and dishes, and manage routing for a structured UI.
* **Task 2: Angular Components, Routing, Services, and Forms**
  + **Usage**: Enhances user experience with structured routing, reusable components, and interactive forms.
  + **Objective**: Create the end-user interface to browse restaurants and dishes with responsive design.
* **Task 3: SQL CRUD Commands, Primary and Foreign Key Relationships**
  + **Usage**: SQL CRUD (Create, Read, Update, Delete) commands are foundational for database operations, and relationships between tables ensure data integrity.
  + **Objective**: Build the database schema, define tables and relationships to support restaurant and dish data management.
* **Task 4: Spring Boot Web Dependency, RestController, RequestMapping**
  + **Usage**: Key backend technologies for building REST APIs, commonly used for scalable web services.
  + **Objective**: Set up backend microservices, define routes for HTTP requests, and handle frontend-backend communication.
* **Task 5: Angular HTTP Client Library, HTTP Request-Response, JSON**
  + **Usage**: Facilitates seamless data exchange between frontend and backend using JSON format, a widely accepted data-interchange format.
  + **Objective**: Use Angular’s HTTP Client for requests to backend APIs, handle responses, and update UI accordingly.