**PROJECT SETUP.**

**1. project Workflow**

* **backend (spring boot)**:
  + created a spring boot project using spring **initializr** : IDE used **Intellij**
  + included dependencies such as **spring web**, **spring data jpa**, **mysql driver** and **spring security** for authentication if needed.
  + configured the database in **application.properties** to connect with a mysql database.
* **frontend (angular)**:
  + set up the angular project using angular cli (ng new food-ordering-app).
  + generated the required components for the app such as home, menu, cart, and checkout using **ng generate component** commands.

**2. database design**

* designed the database schema for entities such as:
  + **product**: for storing food items with fields like productid, name, price, description, and category.
  + **order**: to track customer orders with fields like orderid, customername, orderdate, and totalprice.
  + **orderdetail**: a join table that contains details of each order, including quantity and foreign keys linking to the product and order tables.
* mapped these entities using jpa annotations like **@entity**, **@table**, **@onetomany**, **@manytoone**, etc.

**3. restful api development**

* developed restful endpoints for the backend using spring boot.
  + example endpoints:
    - **productcontroller**: handles requests related to products (e.g., fetching the list of available food items).
    - **ordercontroller**: manages orders placed by customers.
    - **cartcontroller**: manages items in the user's shopping cart.
  + these endpoints interact with the database through services and repositories to perform crud operations.
* example api routes:
  + get /products: fetches all available products.
  + post /orders: places a new order.
  + put /cart: updates the cart items.

**4. frontend development (angular)**

* **components**:
  + built various components to handle user interface elements, such as:
    - menucomponent: displays the list of food items retrieved from the spring boot api.
    - cartcomponent: shows the cart where users can review the items they want to purchase.
    - ordercomponent: handles the checkout and order submission process.
  + used angular services to interact with the backend api for fetching and updating data.
* **routing**:
  + implemented angular routing to navigate between different parts of the app (menu, cart, checkout).
* **forms**:
  + utilized angular forms for handling user inputs like entering customer details during checkout.

**5. data binding and state management**

* implemented two-way data binding to synchronize the data between the frontend and backend.
* used angular services to manage the application’s state, especially for items in the cart, and orders.

**6. service layer (angular & spring boot)**

* **angular services**:
  + created services to interact with the spring boot api using httpclient.
  + example services:
    - productservice: fetches products from the backend.
    - cartservice: manages cart operations like adding or removing items.
    - orderservice: submits orders to the backend.
* **spring boot services**:
  + created service classes in the backend to handle business logic and interact with the repositories.
  + used springs **@service** annotation to define service components.

**7. database integration (mysql )**

* used jpa repositories in spring boot to persist data into the database.
* the entities such as product and order are mapped to database tables using jpa annotations.

**8. security**

* secured the application using spring security or jwt (json web tokens) for user authentication.
* implemented authentication guards in angular to prevent unauthorized access to certain parts of the app (e.g., checkout page)