Reservation System

Overall Purpose: How customer reservations are stored and managed in a restaurant
Customer Table:

- Store information about people making reservations
- Each customer is uniquely identified by customer_id
- PK = Primary Key
- Email must be a unique key
- One RestaurantTable can have multiple reservations (1 -> ∞)

RestaurantTable Table:

- Represents the physical tables in the restaurant
- Each table has a unique table_id
- Includes details like seating capacity and location area
- One RestaurantTable can have many reservations, (1 -> ∞)

Reservation Table:

- Stores each booking made by a customer
- Links both the Customer and the RestaurantTable through foreign keys
- FK = Foreign Key
- Keeps track of who reserved what table and when
- customer_id connects to Customer.customer_id (who make it)
- table_id connects to RestaurantTable.table_id (which table)

Rules:

- No overlapping reservations for the same table/time
- Party size must be less than or equal to the table's capacity
- Each reservation must reference an existing customer and table

Start & Stop

Action	Command / Steps
Run app	./mvnw spring-boot:run or use IntelliJ
Stop app	Ctrl + C in the terminal, or red stop button in IntelliJ
Rebuild after changes	./mvnw clean package then ./mvnw spring-boot:run
Change port	In application-dev.yml → server.port: 8081 (or any free port)

Localhost URLs

Resource	URL
Main app (default)	http://localhost:8080
H2 console	http://localhost:8080/h2-console

Health check *(if actuator added)*	http://localhost:8080/actuator/health
API examples	http://localhost:8080/api/availability (once you add controllers)

Database Console (H2)

JDBC URL	jdbc:h2:mem:resdb
User Name	temp
Password	strongpassword
Console Path	/h2-console (set in YAML)
To access:	run the app → open browser → enter those values → Connect

Profile Management

Environment	Profile	How to activate
Dev (H2)	dev	default in application.yml
MySQL	mysql	SPRING_PROFILES_ACTIVE=mysql ./mvnw spring-boot:run
Postgres	postgres	SPRING_PROFILES_ACTIVE=postgres ./mvnw spring-boot:run

Useful Files

File	Purpose
application.yml	sets the active profile
application-dev.yml	dev-only config (H2 DB, console, etc.)
schema.sql / data.sql	optional scripts that auto-create or seed tables
pom.xml	dependency & project config
ReservationApplication.java	main Spring Boot entry point

When Something Breaks

Symptom	Quick fix
“Site can’t be reached”	app not running → rerun ./mvnw spring-boot:run
Port already in use	lsof -i :8080 → kill -9 <PID>
Old username showing	clear browser autofill on /h2-console
Config not applying	check profile (spring.profiles.active) and restart