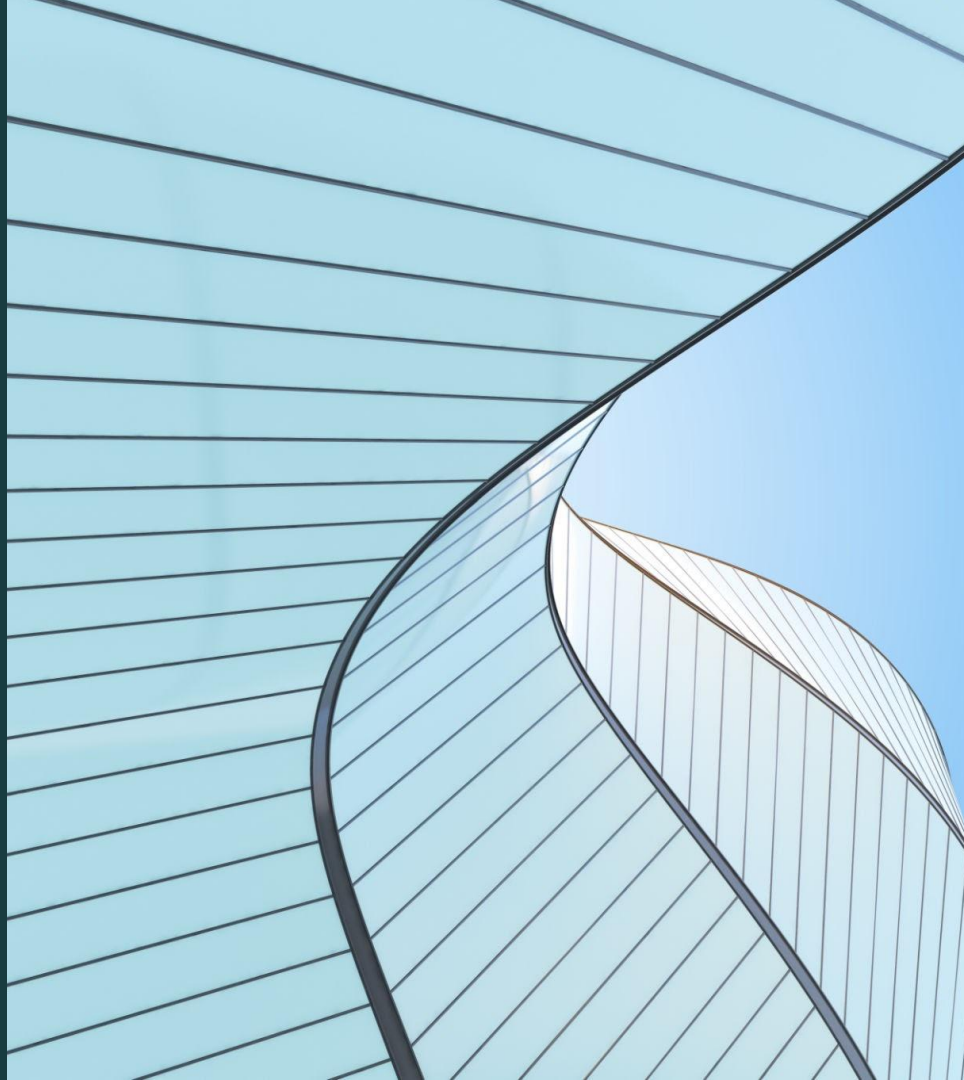
RESTAURANT AUTOMATION SYSTEM PRESENTATION

Melike Aytaç-21070006056

Sultan Esen Murat-21070006039

Yağmur Sabırlı-21070006017

Beren Elçin Polat-22070006064



Introduction to Restaurant Automation System:

Restaurant Automation System is a Java-based desktop application that simplifies daily restaurant operations.
It helps manage orders, menu items, and inventory efficiently using a layered architecture and MySQL database support.
The system improves accuracy and saves time for restaurant staff.

Major Features:

🧾 Order Management
Take and track customer orders efficiently.

🍽️ Menu Management
Add, edit menu items via a user-friendly interface.

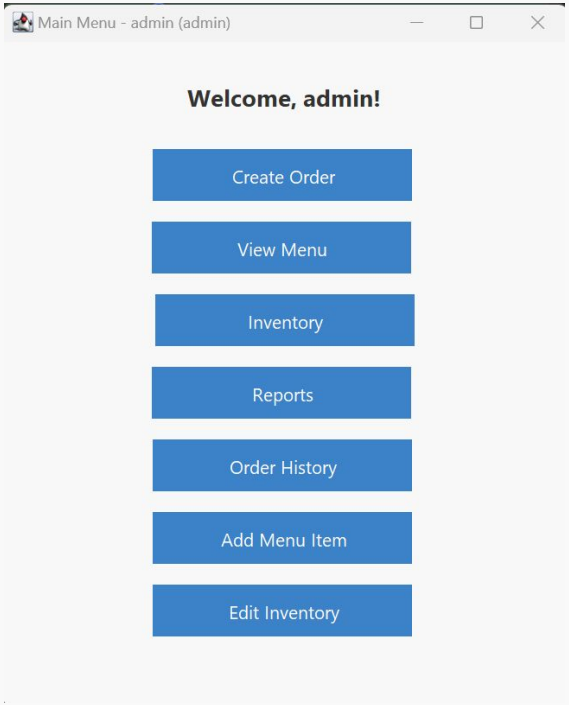
📦 Inventory Tracking
Monitor stock levels and automatically update after each order.

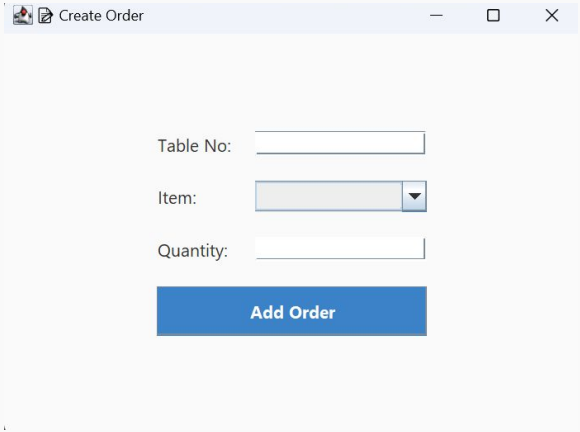
📊 Sales Reporting
Generate daily reports for total orders, and earnings.

🔐 User Login System
Simple login screen to control system access.

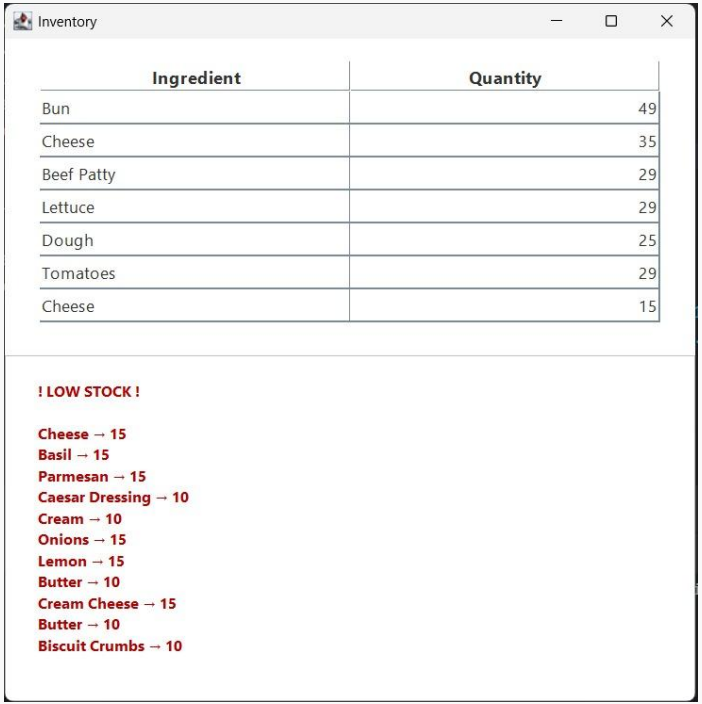
💾 Database Integration
Connects with MySQL to store menu, order, and inventory data.

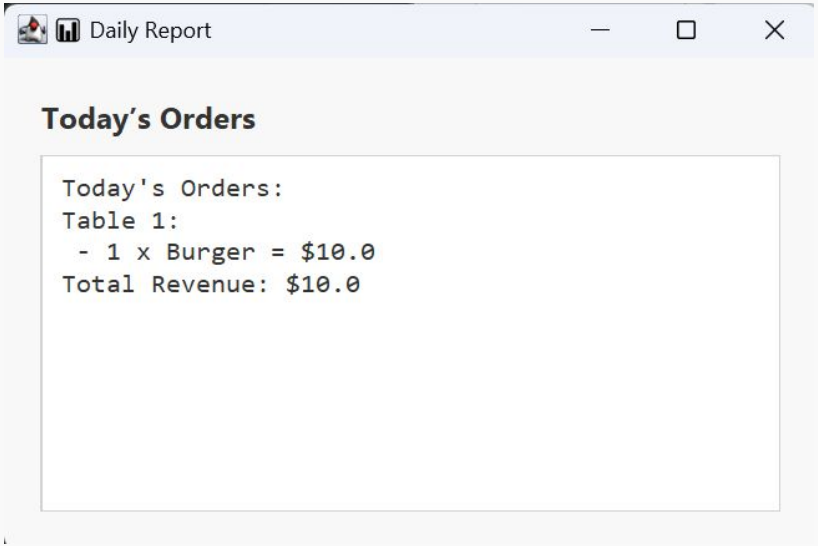
App Features:



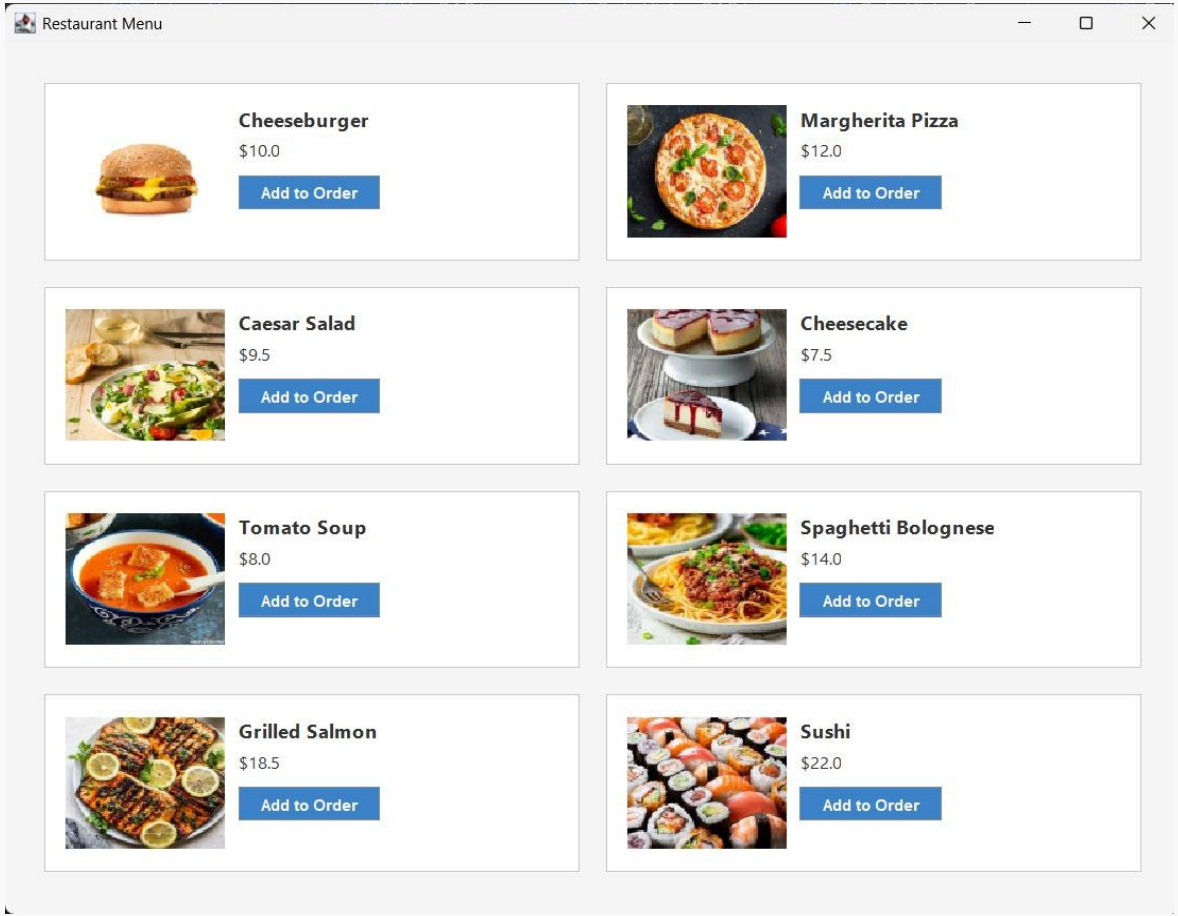


More App Features:

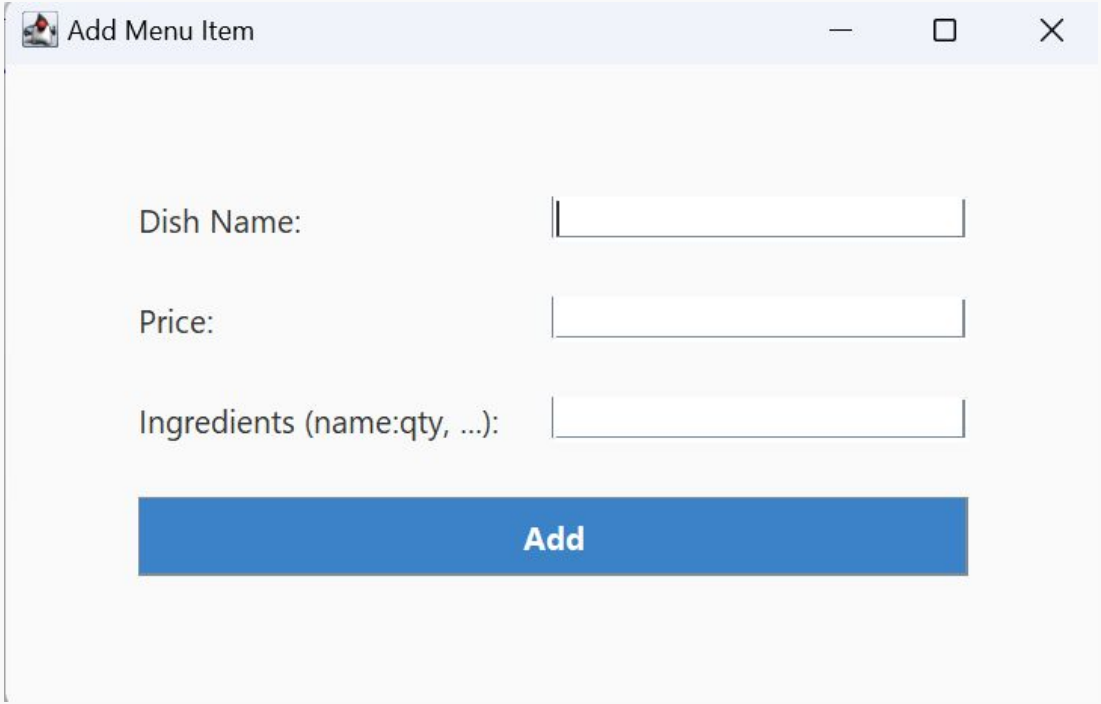


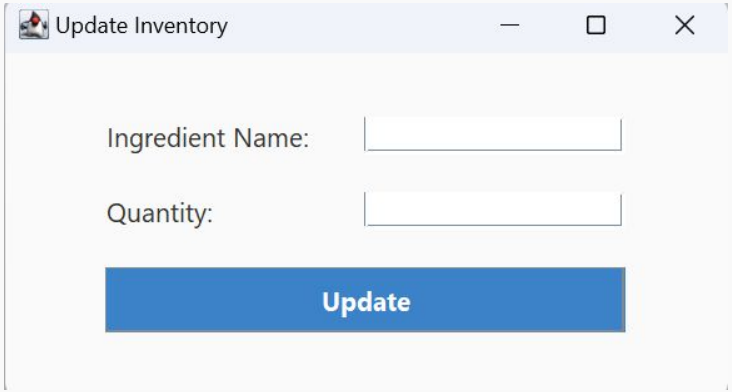


More App Features:

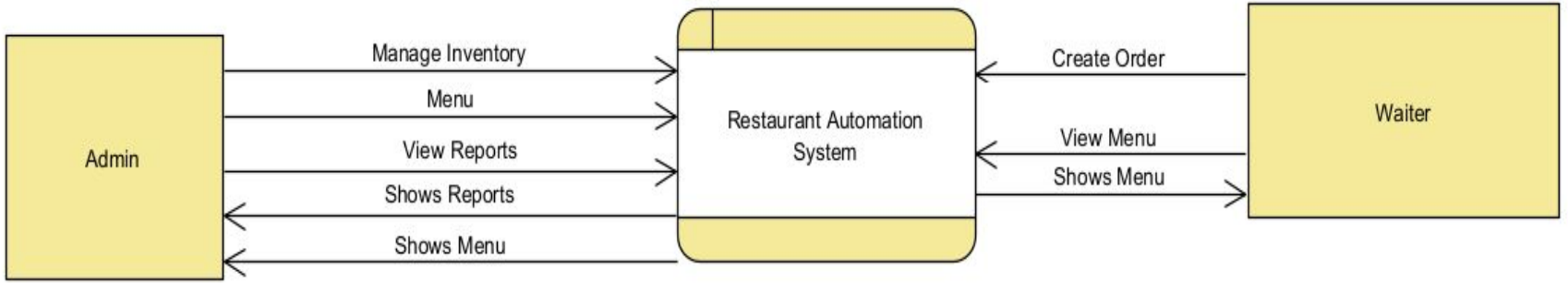


More App Features:

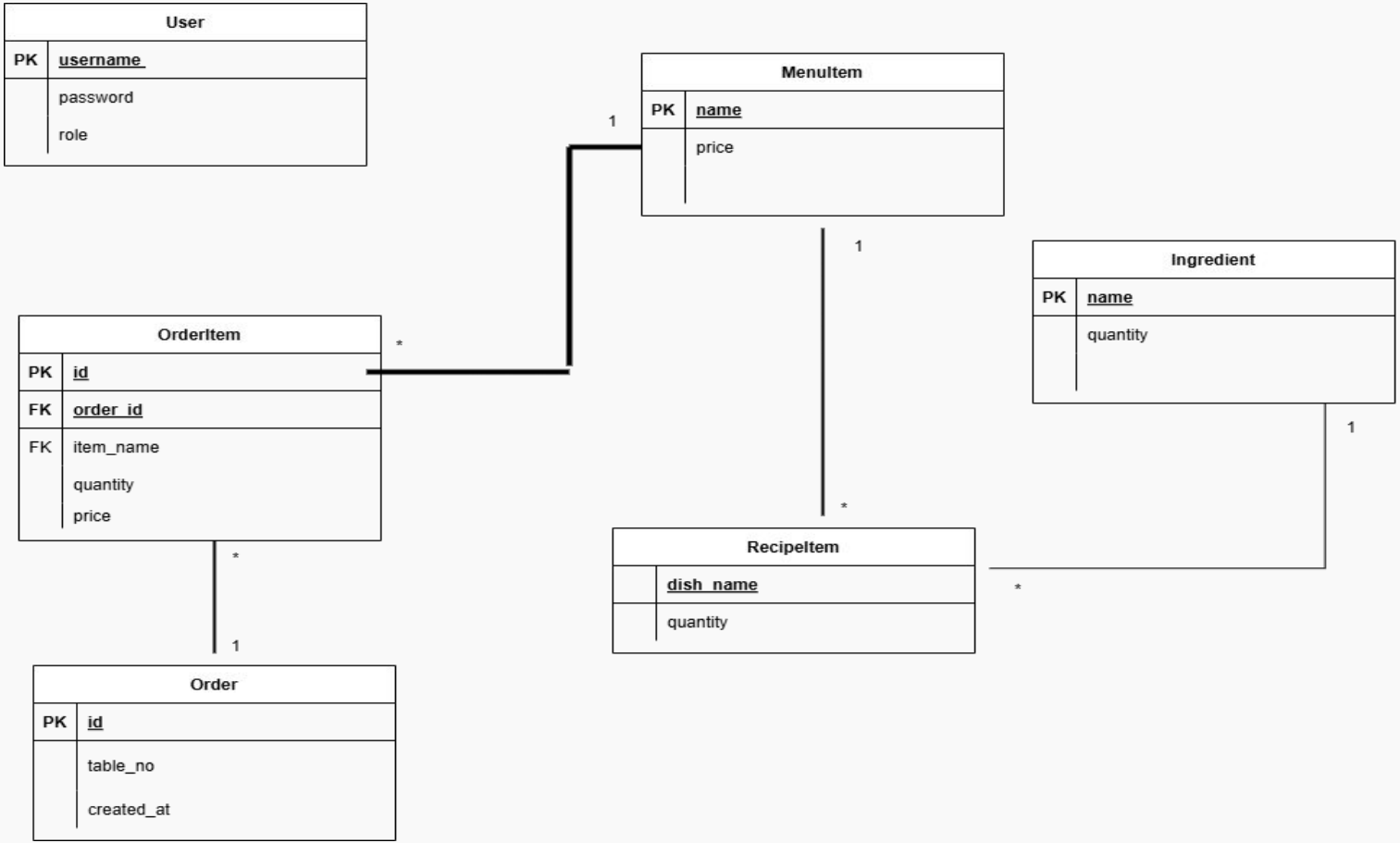




Context Diagram



ER DIAGRAM


User
PK
username
password
role
MenuItem
PK
name
price
1
1
Ingredient
PK
name
quantity
1
OrderItem
PK
id
FK
order_id
FK
item_name
quantity
price
*
*
1
*
RecipeItem
dish_name
quantity
*
Order
PK
id
table_no
created_at

Some Test Cases:

🧪 Test Case 1 — Go to “Login” page and Enter admin PIN .

🧪 Test Case 2 — Go to “Login” page and Enter worker PIN

🧪 Test Case 3 — Go to “Menu” page and Display the current menu

🧪 Test Case 4 — Go to “Report” page and Display the current report

🧪Test Case 5 — Go to database and Add a new Menu Item via code, then display it from Menu page.

🧪Test Case 6 — Go to Inventory and Add a new ingredient, then deduct it and check if updated.

🧪Test Case 7 — Add a Recipe (Dish + Ingredient), then fetch ingredients of the Dish via code.

🧪Test Case 8 — Save a new Order from backend (OrderDAO), then check order_items table to verify.

🧪Test Case 9 — From backend, call fetchAllOrders() from OrderHistoryDAO and confirm recent orders are shown.

Test Case 8 -- Save a new Order from backend (OrderDAO), then check order_items table to verify.

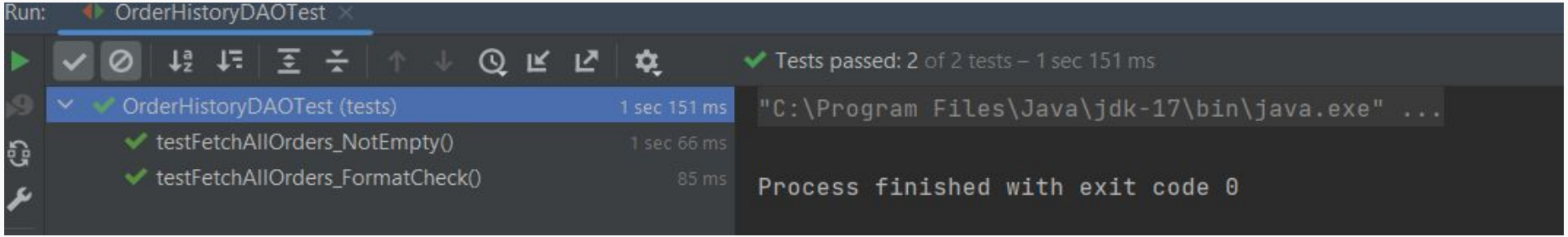
Result Grid | Filter Rows: | Edit:

id	order_id	item_name	quantity	price
1	1	Tomato Soup	1	10
2	2	Tomato Soup	1	10
3	3	Tomato Soup	1	10
4	4	Cheeseburger	1	10
5	5	Cheeseburger	1	10
6	6	Cheeseburger	1	10
7	7	Tomato Soup	1	10
8	8	Cheeseburger	1	10
9	9	Cheeseburger	1	10
10	10	Cheeseburger	1	10
11	11	Cheeseburger	2	50
NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

id	order_id	item_name	quantity	price
1	1	Tomato Soup	1	10
2	2	Tomato Soup	1	10
3	3	Tomato Soup	1	10
4	4	Cheeseburger	1	10
5	5	Cheeseburger	1	10
6	6	Cheeseburger	1	10
7	7	Tomato Soup	1	10
8	8	Cheeseburger	1	10
9	9	Cheeseburger	1	10
10	10	Cheeseburger	1	10
11	11	Cheeseburger	2	50
12	12	UnitTestItem	2	10
NULL	NULL	NULL	NULL	NULL

Test Case 9 — From backend, call fetchAllOrders() from OrderHistoryDAO and confirm recent orders are shown.



Test Case 7 — Add a Recipe (Dish + Ingredient), then fetch ingredients of the Dish via code.

Result Grid | Filter Rows: | Edit:

id	dish_name	ingredient_name	quantity
1	Cheeseburger	Bun	1
2	Cheeseburger	Beef Patty	1
3	Cheeseburger	Cheese	1
4	Cheeseburger	Lettuce	1
NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

id	dish_name	ingredient_name	quantity
1	Cheeseburger	Bun	1
2	Cheeseburger	Beef Patty	1
3	Cheeseburger	Cheese	1
4	Cheeseburger	Lettuce	1
8	TestDish_Unit	TestIngredient_Unit	3
NULL	NULL	NULL	NULL

Some Potential Risks:

Testing Risks

🧾Requirements

The test plan and test schedule are based on the current Requirements Document. Any changes to the requirements could affect the test schedule.

⏰ Time

The schedule for each phase is very aggressive and could affect testing. A delay in one of the other phases could result in a subsequent delay in the test phase. Project management support is required to reduce the risk and meet the forecasted completion date.

🖥️Environment

Missing/unstable local MySQL setup could hinder integration testing

👥Personnel

All developers should be familiar with the testing tools. Unexpected turnovers can impact the schedule. All developers will also need the following resources available: developers and test users.

Some Constraints:

🔐 Only authorized users (e.g., waitstaff or manager) can log in

📶 Requires active database connection for all operations

⚙️ System runs only on desktop environments (Java SE required)

Cocomo Analysis:

Function Point Calculator	Total	Factor	FP
The Madison Utilities, Department of Computer Science, James Madison University	106	0.83	**88**

Direct Measure	Count Simple	Average	Complex	Weighted Measure
External Inputs (EIs)	2	2	1	20
External Outputs (EOs)	2	2	1	25
External Inquiries (EQs)	1	1	0	7
Internal Logical Files (ILFs)	2	2	1	49
External Interface Files (EIFs)	1	0	0	5

Clear

Value Adjustment Factor	0	1	2	3	4	5
The system requires reliable backup and recovery.	○	◉	○	○	○	○
Specialized data communications are required.	○	◉	○	○	○	○
There are distributed processing functions.	◉	○	○	○	○	○
Performance is critical.	○	◉	○	○	○	○
The system runs in an existing, heavily utilized operational environment.	◉	○	○	○	○	○
The system requires on-line data entry.	◉	○	○	○	○	○
The on-line data entry requires transactions over multiple screens/operations.	◉	○	○	○	○	○
ILFs are updated on-line.	◉	○	○	○	○	○
The inputs, outputs, files or inquiries are complex.	○	○	◉	○	○	○
The internal processing is complex.	○	○	◉	○	○	○
The code is designed to be reusable.	○	○	○	○	◉	○
Conversions /installation are included in the design.	○	○	◉	○	○	○
The system is designed for multiple installations in different organizations.	◉	○	○	○	○	○
The system is designed to facilitate change and ease of use.	○	○	○	○	○	◉

Cocomo Calculator

QSM
Quantitative Software Management

DEMO BLOG CONTACT

PROBLEMS WE SOLVE PRODUCTS CONSULTING TRAINING RESOU

* Languages with updated gearing factors.
+ New languages for which gearing factor data was not previously reported.

QSM SLOC/FP Data				
Language	**Avg**	**Median**	**Low**	**High**
ABAP (SAP) *	28	18	16	60
ASP*	51	54	15	69
Assembler *	119	98	25	320
Brio +	14	14	13	16
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
COBOL *	61	55	23	297
Cognos Impromptu Scripts +	47	42	30	100
Cross System Products (CSP) +	20	18	10	38
Cool:Gen/IEF *	32	24	10	82
Datastage	71	65	31	157
Excel *	209	191	131	315
Focus *	43	45	45	45
FoxPro	36	35	34	38
HTML *	34	40	14	48
J2EE *	46	49	15	67
Java *	53	53	14	134

Mike's Basic COCOMO Calculator!

Enter the number of estimated lines of code and the calculator will determine how much time and how many people will be needed!

4 Thousands of Lines of Estimated Code.

Perform Calculation

Organic Values

Number of Months Needed: 6.0623069 Number of People Needed: 2

SemiDetached Values

Number of Months Needed: 6.3232443 Number of People Needed: 2

Embedded Values

Number of Months Needed: 6.4142789 Number of People Needed: 3

What We Have Done So Far:

Task Name	Duration	Start	Finish	Predecessors
Restaurant Automation System	**38 days**	**Tue 15.04.25**	**Wed 4.06.25**	
Iteration 1	**20 days**	**Tue 15.04.25**	**Sun 11.05.25**	
Backend Functionality	18 days	Tue 15.04.25	Thu 8.05.25	
Data Modelling	2 days	Fri 9.05.25	Sun 11.05.25	3
Iteration 2	**12 days**	**Sun 11.05.25**	**Mon 26.05.25**	**2**
Interface Coding	8 days	Sun 11.05.25	Tue 20.05.25	
Storage Integration	4 days	Fri 9.05.25	Tue 13.05.25	3
Iteration 3	**8 days**	**Mon 26.05.25**	**Wed 4.06.25**	**5**
Interface Improvements	3 days	Wed 21.05.25	Fri 23.05.25	6
Integration Testing	3 days	Mon 26.05.25	Wed 28.05.25	9
Bug Fixing	4 days	Thu 29.05.25	Tue 3.06.25	10