Technology Trend REPORT

Distributed Systems

Software Design with AI in Cloud 4th year Student:
Ivan Lapickij

Lecturer: Peter Vargovcik

16.04.2025

Technological University of Shannon Athlone

INTRODUCTION

Technology Trend is my 4th year project for Distributed Systems. The idea is to capture all trending technology of over a year with as much data like price, year, company as possible.These days technology is advancing very fast and it's hard to keep track of what is new. The Technology Trend project is responsible for storing the newest technology data so it would be easier to track between years , price range and categories. Aswell if you made a mistake or something changed , don't worry - this application can do all CRUD operations. Further I'm going to explain processes involved in creating this application.

OBJECTIVES

Using Java API for RESTful Web Services (Jax-Rs):

- ★ 1. Build a client application that sends all of the HTTP requests: *GET/PUT/POST/DELETE.*
- ★ 2. Build a server application using tomcat server, that responds to all of the HTTP requests: *GET/PUT/POST/DELETE*
- ★ 3. The client application will parses the response using XMLPullParser and outputs to the GUI" + "A tomcat server that responds to all of the HTTP requests: *GET/PUT/POST/DELETE"*
- ★ 4. The data in the response will be taken from an HSQLDB database.

ARCHITECTURE DIAGRAM

The architecture diagram illustrates the overall structure and interaction between the core components of the distributed system project. It provides a clear view of how the frontend, backend, database, and server communicate to deliver a full RESTful application.

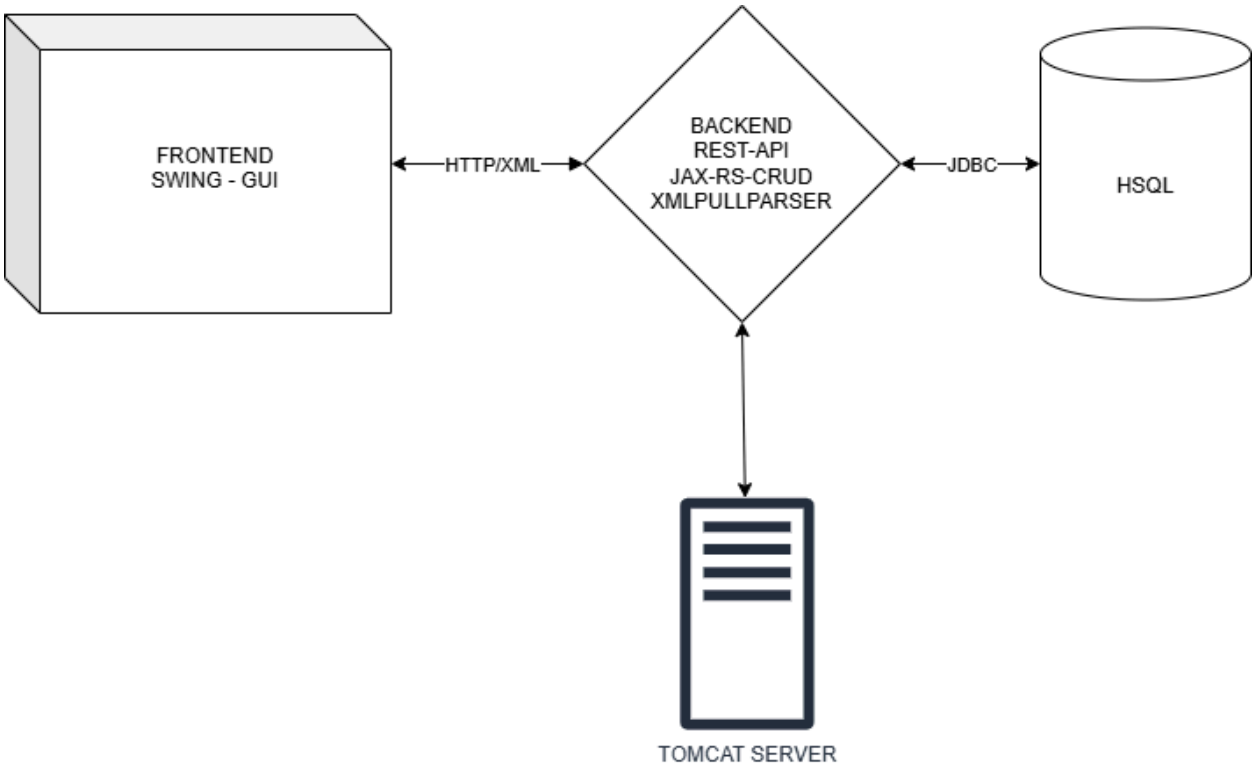


Figure 1. Illustrating architecture diagram of application.

➔ **Database Layer (HSQLDB with ANT)**

The project uses HSQLDB (HyperSQL Database), an in-memory and lightweight relational database ideal for embedded Java applications.

An ANT build script was used to automate the creation and population of the database tables with initial data.

The ProductDAO and related classes connect to the HSQLDB using JDBC to perform SQL operations such as SELECT, INSERT, UPDATE, and DELETE.

- SQL Queries.

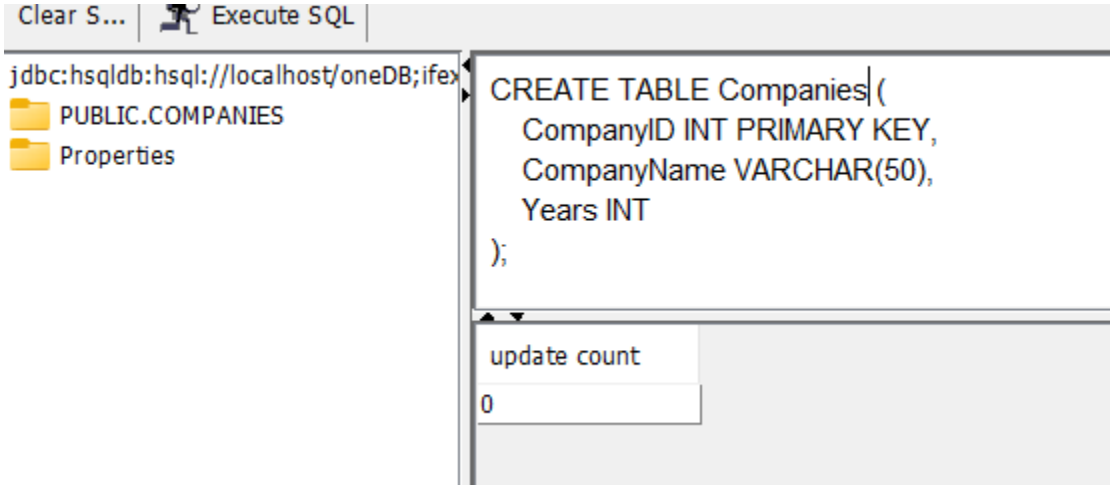


Figure 2. Creates Companies Table.

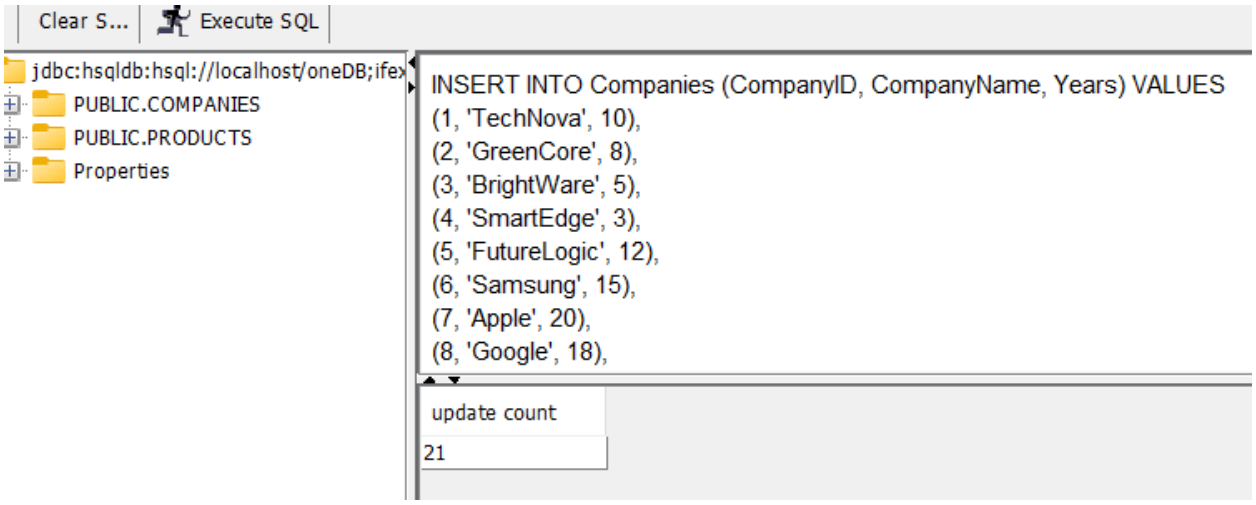


Figure 3. Populates Companies Table.

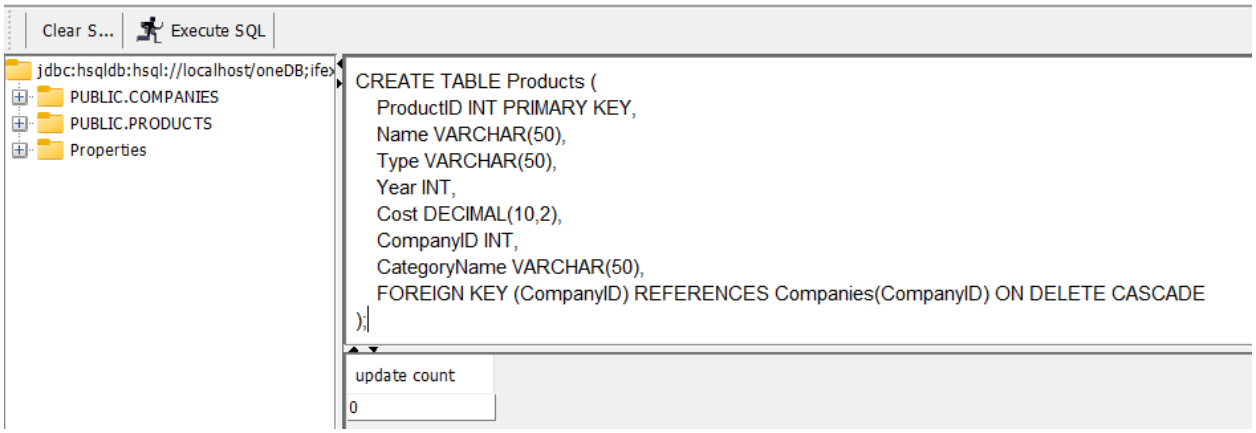


Figure 4. Creates Products Table.

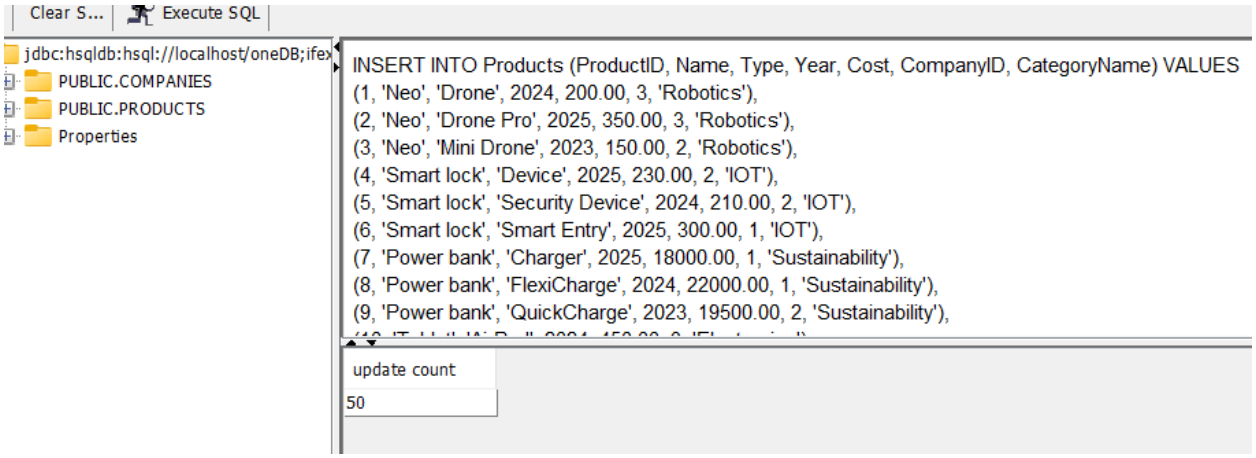


Figure 5. Populates Products Table.

➔ **Backend Layer (Java REST API using JAX-RS)**

The backend is built using JAX-RS, a Java API for building RESTful web services.

The application is deployed on a Tomcat server, which exposes REST endpoints to handle HTTP methods:

GET – fetch all products or filter by ID/name

POST – add a new product

PUT – update product details (e.g., price)

DELETE – remove individual or all products

Responses are provided in XML format using JAXB, making them compatible with the frontend's XML parser.

- **Wrapper allows JAXB to parse XML like this**

  <products>

  <product>...</product>

  <product>...</product>

  </products>

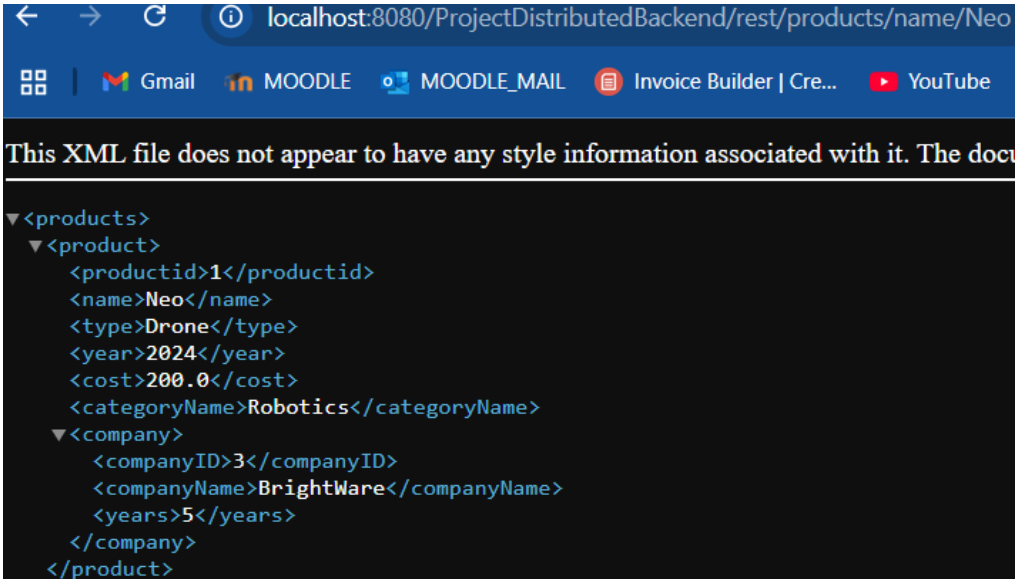


Figure 6. Showing Endpoint for getting ProductsByName with XML response.

➔ Frontend Layer (Java Swing GUI)

The frontend is developed using Java Swing, offering a user-friendly GUI.

The left side contains a control panel with buttons for each CRUD operation:

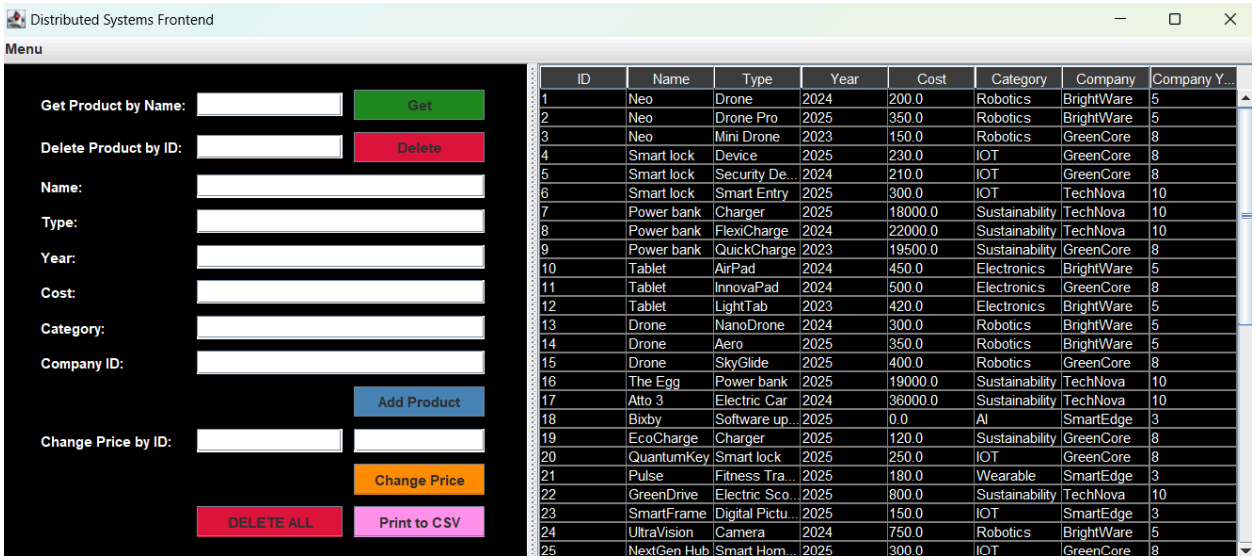


Figure 7. Final GUI showing CRUD controls on the left and data table on the right.

Add Product, Get Product by Name, Delete by ID, Update Price, Delete All, Export to CSV

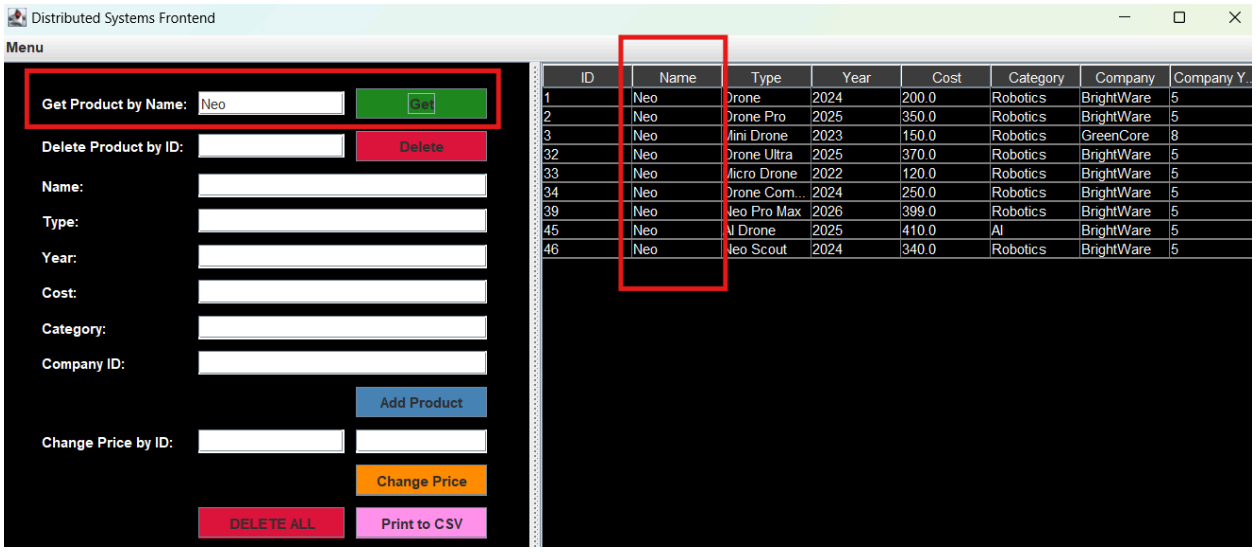


Figure 8. Showing working functionality of getting ProductsByName from DB .

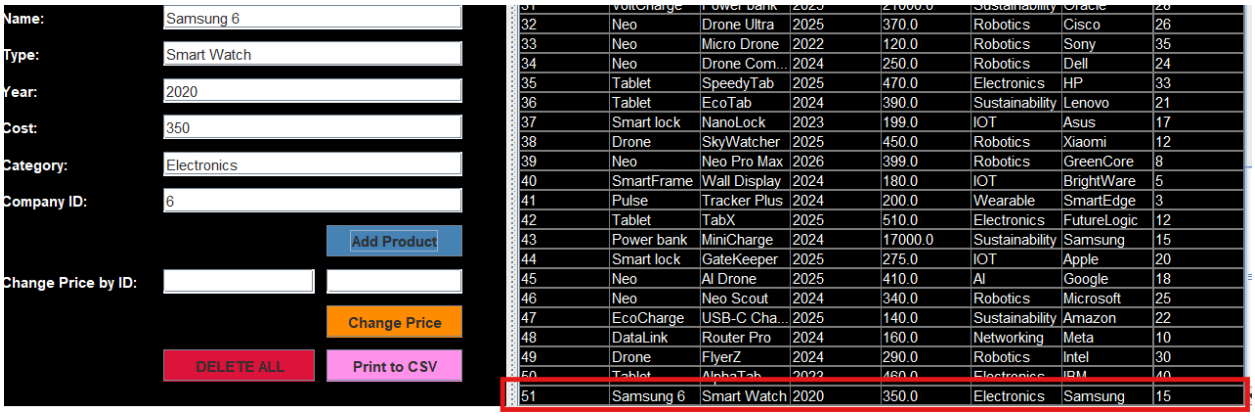


Figure 9. Illustrating addProduct functionality working.

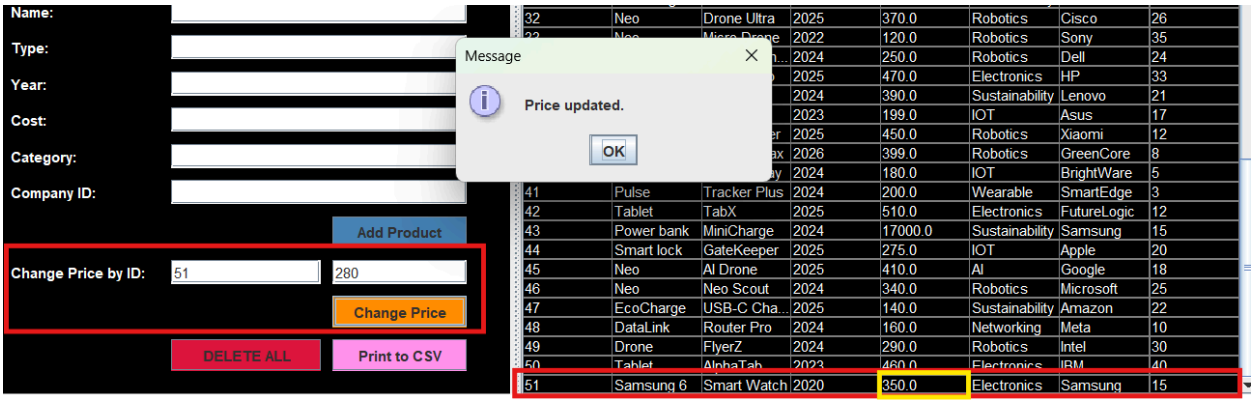


Figure 10. Image displays data before price change functionality.

Figure 11. Image displays data after price change functionality.

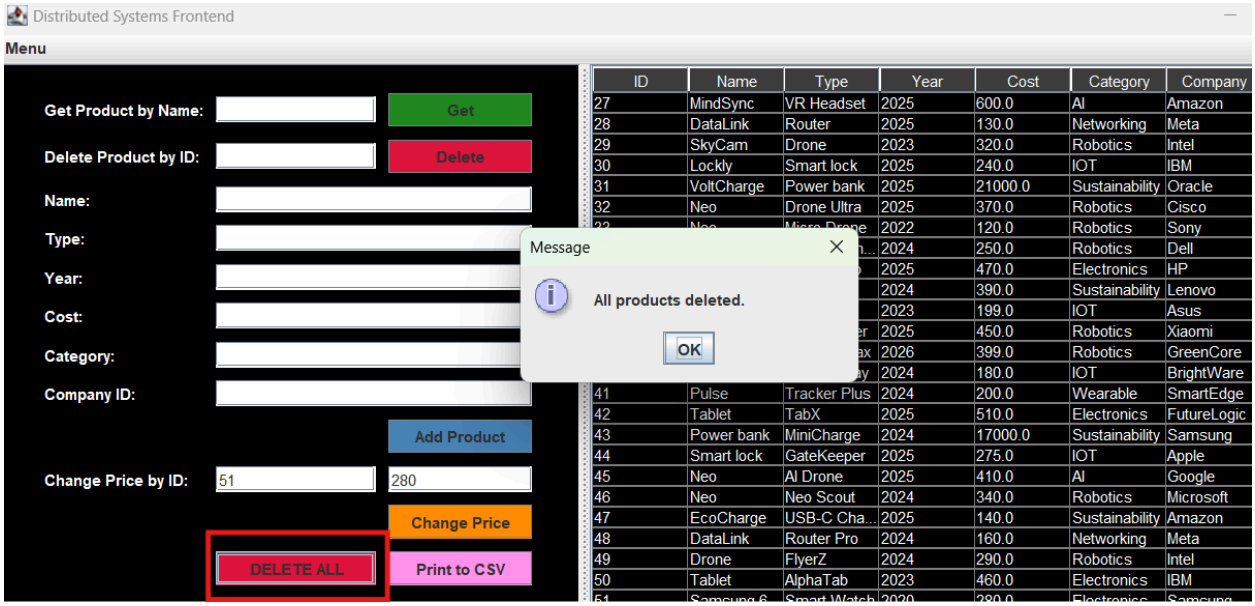


Figure 12. Displayed data before DELETE ALL functionality takes effect.

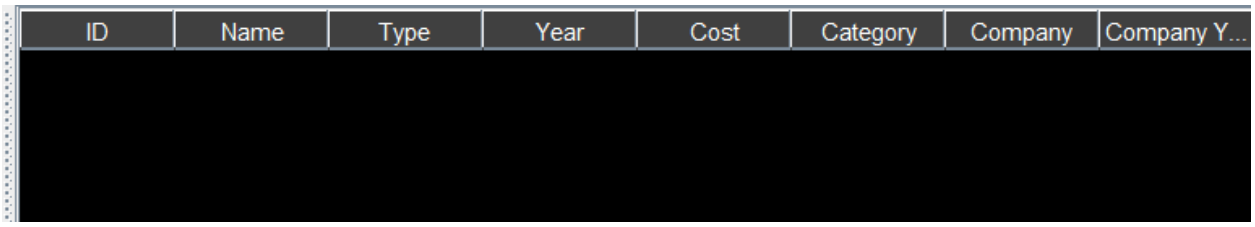


Figure 13. Displayed data after DELETE ALL functionality takes effect.

Figure 14. Displayed data before DELETE By ID functionality takes effect.

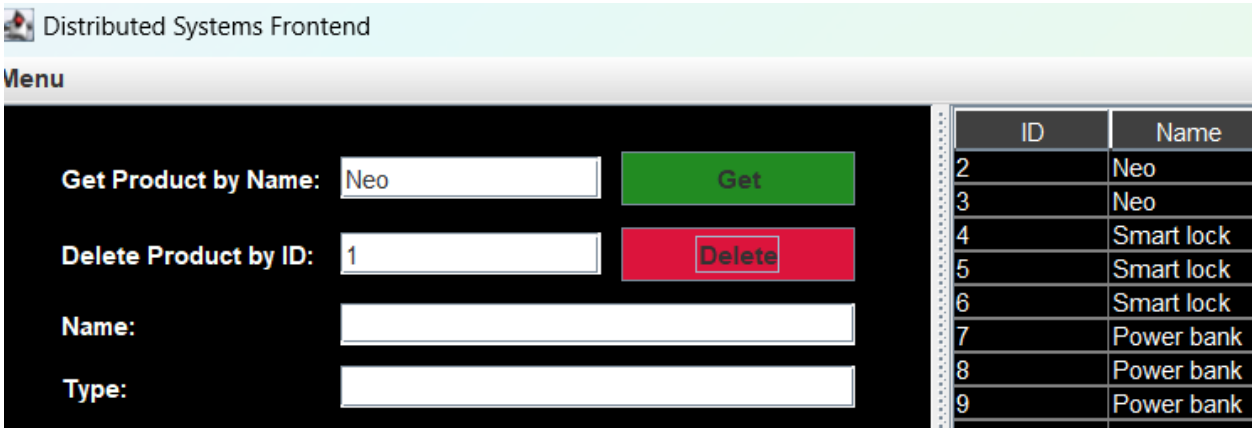


Figure 14. Displayed data after DELETE By ID functionality takes effect.

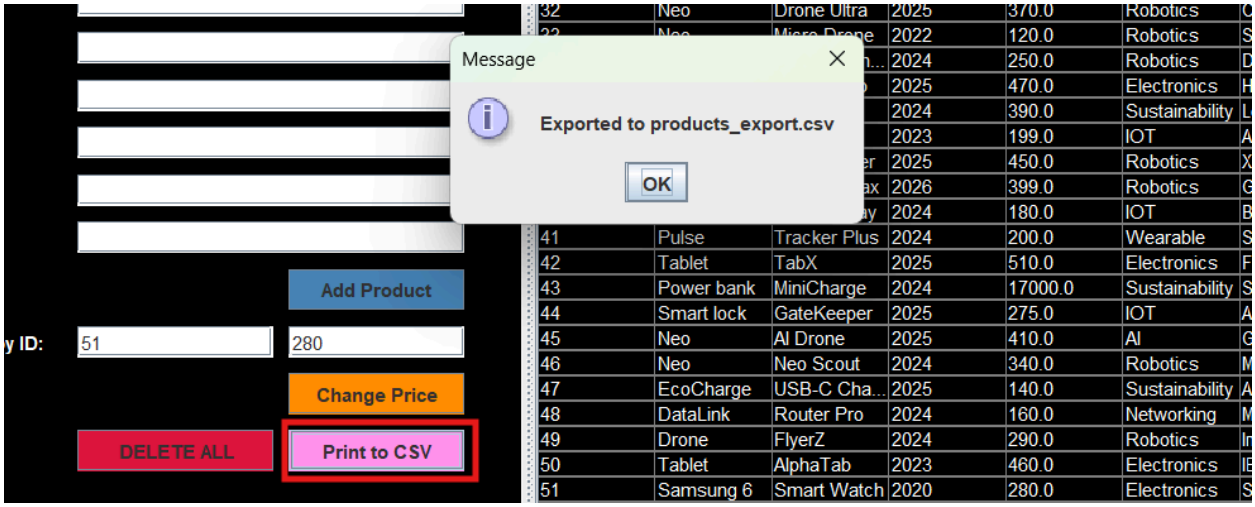
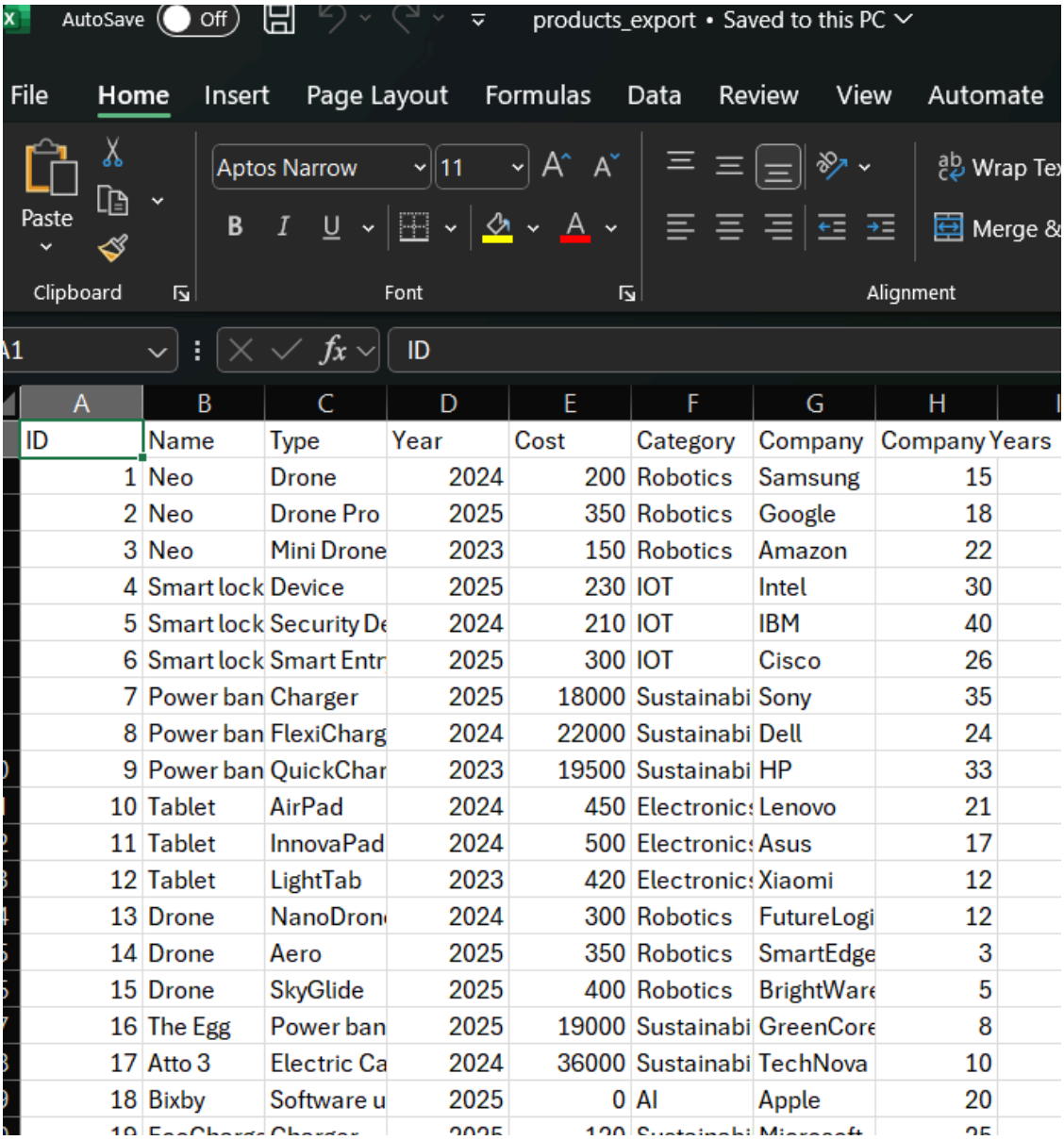


Figure 15. Illustrates feedback of 'Export to CSV' functionality.



ID	Name	Type	Year	Cost	Category	Company	Company Years
1	Neo	Drone	2024	200	Robotics	Samsung	15
2	Neo	Drone Pro	2025	350	Robotics	Google	18
3	Neo	Mini Drone	2023	150	Robotics	Amazon	22
4	Smart lock	Device	2025	230	IOT	Intel	30
5	Smart lock	Security De	2024	210	IOT	IBM	40
6	Smart lock	Smart Entr	2025	300	IOT	Cisco	26
7	Power ban	Charger	2025	18000	Sustainabi	Sony	35
8	Power ban	FlexiCharg	2024	22000	Sustainabi	Dell	24
9	Power ban	QuickChar	2023	19500	Sustainabi	HP	33
10	Tablet	AirPad	2024	450	Electronic:	Lenovo	21
11	Tablet	InnovaPad	2024	500	Electronic:	Asus	17
12	Tablet	LightTab	2023	420	Electronic:	Xiaomi	12
13	Drone	NanoDron	2024	300	Robotics	FutureLogi	12
14	Drone	Aero	2025	350	Robotics	SmartEdge	3
15	Drone	SkyGlide	2025	400	Robotics	BrightWare	5
16	The Egg	Power ban	2025	19000	Sustainabi	GreenCore	8
17	Atto 3	Electric Ca	2024	36000	Sustainabi	TechNova	10
18	Bixby	Software u	2025	0	AI	Apple	20

Figure 16. Illustrates proof of 'Export to CSV' functionality.

Each CRUD action sends an appropriate HTTP request to the backend and refreshes the displayed data accordingly.

CONCLUSION

In this project, I make one full application with frontend and backend. Frontend use Java Swing and backend use REST API. I connect them with HTTP and XML. Database is HSQL and I use it for saving products and companies. Everything work together. I learn how to use XMLPullParser, how to send request like GET, POST, PUT, DELETE. I see how client and server talk in real system. It was good project for learning distributed system.

REFERENCES

HSQL queries structure

The Egg - Enron Corporation

Lockly

The BYD ATTO 3 Electric Family SUV Now Available in Ireland

Buy DJI Neo (No RC)

Bixby | Apps & Services | Samsung US

APPENDIX

```
-- Create the Companies table with a foreign key linking to Products.
CREATE TABLE Companies (
    CompanyID INT PRIMARY KEY,
    CompanyName VARCHAR(50),
    Years INT
);

-- Insert records into Companies.
INSERT INTO Companies (CompanyID, CompanyName, Years) VALUES
(1, 'TechNova', 10),
(2, 'GreenCore', 8),
(3, 'BrightWare', 5),
(4, 'SmartEdge', 3),
(5, 'FutureLogic', 12),
(6, 'Samsung', 15),
(7, 'Apple', 20),
(8, 'Google', 18),
(9, 'Microsoft', 25),
(10, 'Amazon', 22),
(11, 'Meta', 10),
(12, 'Intel', 30),
(13, 'IBM', 40),
(14, 'Oracle', 28),
(15, 'Cisco', 26),
(16, 'Sony', 35),
(17, 'Dell', 24),
(18, 'HP', 33),
(19, 'Lenovo', 21),
(20, 'Asus', 17),
(21, 'Xiaomi', 12);



-- Create the Products table
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(50),
    Type VARCHAR(50),
    Year INT,
    Cost DECIMAL(10,2),
```

```
    CompanyID INT,
    CategoryName VARCHAR(50),
    FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID) ON DELETE
CASCADE
);


-- Insert records into Products.
INSERT INTO Products (ProductID, Name, Type, Year, Cost, CompanyID, CategoryName)
VALUES
(1, 'Neo', 'Drone', 2024, 200.00, 6, 'Robotics'),
(2, 'Neo', 'Drone Pro', 2025, 350.00, 8, 'Robotics'),
(3, 'Neo', 'Mini Drone', 2023, 150.00, 10, 'Robotics'),
(4, 'Smart lock', 'Device', 2025, 230.00, 12, 'IOT'),
(5, 'Smart lock', 'Security Device', 2024, 210.00, 13, 'IOT'),
(6, 'Smart lock', 'Smart Entry', 2025, 300.00, 15, 'IOT'),
(7, 'Power bank', 'Charger', 2025, 18000.00, 16, 'Sustainability'),
(8, 'Power bank', 'FlexiCharge', 2024, 22000.00, 17, 'Sustainability'),
(9, 'Power bank', 'QuickCharge', 2023, 19500.00, 18, 'Sustainability'),
(10, 'Tablet', 'AirPad', 2024, 450.00, 19, 'Electronics'),
(11, 'Tablet', 'InnovaPad', 2024, 500.00, 20, 'Electronics'),
(12, 'Tablet', 'LightTab', 2023, 420.00, 21, 'Electronics'),
(13, 'Drone', 'NanoDrone', 2024, 300.00, 5, 'Robotics'),
(14, 'Drone', 'Aero', 2025, 350.00, 4, 'Robotics'),
(15, 'Drone', 'SkyGlide', 2025, 400.00, 3, 'Robotics'),
(16, 'The Egg', 'Power bank', 2025, 19000.00, 2, 'Sustainability'),
(17, 'Atto 3', 'Electric Car', 2024, 36000.00, 1, 'Sustainability'),
(18, 'Bixby', 'Software update', 2025, 0.00, 7, 'AI'),
(19, 'EcoCharge', 'Charger', 2025, 120.00, 9, 'Sustainability'),
(20, 'QuantumKey', 'Smart lock', 2025, 250.00, 11, 'IOT'),
(21, 'Pulse', 'Fitness Tracker', 2025, 180.00, 14, 'Wearable'),
(22, 'GreenDrive', 'Electric Scooter', 2025, 800.00, 5, 'Sustainability'),
(23, 'SmartFrame', 'Digital Picture Frame', 2025, 150.00, 6, 'IOT'),
(24, 'UltraVision', 'Camera', 2024, 750.00, 7, 'Robotics'),
(25, 'NextGen Hub', 'Smart Home Hub', 2025, 300.00, 8, 'IOT'),
(26, 'SonicDrive', 'Electric Car', 2024, 40000.00, 9, 'Sustainability'),
(27, 'MindSync', 'VR Headset', 2025, 600.00, 10, 'AI'),
(28, 'DataLink', 'Router', 2025, 130.00, 11, 'Networking'),
(29, 'SkyCam', 'Drone', 2023, 320.00, 12, 'Robotics'),
(30, 'Lockly', 'Smart lock', 2025, 240.00, 13, 'IOT'),
(31, 'VoltCharge', 'Power bank', 2025, 21000.00, 14, 'Sustainability'),
(32, 'Neo', 'Drone Ultra', 2025, 370.00, 15, 'Robotics'),
(33, 'Neo', 'Micro Drone', 2022, 120.00, 16, 'Robotics'),
```

(34, 'Neo', 'Drone Compact', 2024, 250.00, 17, 'Robotics'),
(35, 'Tablet', 'SpeedyTab', 2025, 470.00, 18, 'Electronics'),
(36, 'Tablet', 'EcoTab', 2024, 390.00, 19, 'Sustainability'),
(37, 'Smart lock', 'NanoLock', 2023, 199.00, 20, 'IOT'),
(38, 'Drone', 'SkyWatcher', 2025, 450.00, 21, 'Robotics'),
(39, 'Neo', 'Neo Pro Max', 2026, 399.00, 2, 'Robotics'),
(40, 'SmartFrame', 'Wall Display', 2024, 180.00, 3, 'IOT'),
(41, 'Pulse', 'Tracker Plus', 2024, 200.00, 4, 'Wearable'),
(42, 'Tablet', 'TabX', 2025, 510.00, 5, 'Electronics'),
(43, 'Power bank', 'MiniCharge', 2024, 17000.00, 6, 'Sustainability'),
(44, 'Smart lock', 'GateKeeper', 2025, 275.00, 7, 'IOT'),
(45, 'Neo', 'AI Drone', 2025, 410.00, 8, 'AI'),
(46, 'Neo', 'Neo Scout', 2024, 340.00, 9, 'Robotics'),
(47, 'EcoCharge', 'USB-C Charger', 2025, 140.00, 10, 'Sustainability'),
(48, 'DataLink', 'Router Pro', 2024, 160.00, 11, 'Networking'),
(49, 'Drone', 'FlyerZ', 2024, 290.00, 12, 'Robotics'),
(50, 'Tablet', 'AlphaTab', 2023, 460.00, 13, 'Electronics');

-- Display records from Products
SELECT * FROM Products;

-- Display records from Companies
SELECT * FROM Companies;